EGG-SSRE-9283

SCDAP/RELAP5 PROGRAMMERS MANUAL

E. W. Coryell
L.J. Siefken
R.J. Wagner

Published May 1994

EG&G Idaho, Inc.
Idaho National Engineering Laboratory
Idaho Falls, ID 83415

Prepared for the
Division of Systems Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555
Under DOE Idaho Field Office
Contract No. DE-AC07-76IDO1570

ABSTRACT

The SCDAP/RELAP5 code has reached the point when user's external to the Idaho National Engineering Laboratory (INEL) are extensively modifying the code. In order to make the task of modifying the code easier for the external developer, and to make the task of maintaining the code easier for INEL personnel, a programmer's manual has been developed. This manual will allow INEL code developers to document the diverse set of utilities and programming tools used to develop SCDAP/RELAP5 as well as documenting a set of programming guidelines which we have found to be useful.

SUMMARY

The severe accident transient analysis code, SCDAP/RELAP5, has been developed at the Idaho National Engineering Laboratory (INEL) for the U.S. Nuclear Regulatory Commission (NRC) to provide an advanced best-estimate predictive capability for use in severe accident applications in support of the regulatory process. Code uses included analysis required to support rulemaking, licensing audit calculations, evaluation of accident mitigation strategies, and experiment planning and analysis. Specific applications of this capability have included analytical support for the Loss-of-Fluid-Test (LOFT), Power Burst Facility (PBF), ACRR, MIST, ROSA IV, and NRU experimental programs, as well as simulations of transients that lead to severe accidents, such as loss of coolant, anticipated transients without scram (ATWS), and operational transients in reactor systems.

SCDAP/RELAP5 was developed by integrating three separate codes, RELAP5, SCDAP, and COUPLE. These codes were combined to model the coupled interactions that occur between the core, the coolant system, and the debris beds which can form during a severe accident. For example, blockage in the core, caused by fuel assembly meltdown, can have a significant effect on coolant system flows. Heat transfer between a debris bed formed in the lower plenum and emergency system fluid can significantly affect the coolant conditions entering the core. These and many other coupled effects can have a significant effect on the release of fission products from the reactor system.

One of the important aspects of maintaining a detailed code, such as SCDAP/RELAP5, is the incorporation of new information gained through severe accident experiments, analytic programs, and user experience. In many cases new models or code improvements are being developed by researchers external to the INEL. As a result, a set of programming guidelines have been developed to facilitate the incorporation of the improvements and to insure that the code still maintains the same functionality on a wide variety of computer platforms. This manual is intended to provide both internal and external developers with a resource to document our experience and understanding of the code.

Table of Contents

Table of Contents

List Of Figures

List Of Tables

SCDAP/RELAP5/MOD3.1[a] Programmer's Manual

1. INTRODUCTION

The SCDAP/RELAP5 computer code is designed to describe the overall reactor coolant system (RCS) thermal-hydraulic response, core damage progression, and, in combination with VICTORIA,[1] fission product release and transport during severe accidents. The code is being developed at the Idaho National Engineering Laboratory (INEL) under the primary sponsorship of the Office of Nuclear Regulatory Research of the U.S. Nuclear Regulatory Commission (NRC).

1.1 General Code Capabilities

The code is the result of merging the RELAP5/MOD3[2] and SCDAP[3] models. The RELAP5 models calculate the overall RCS thermal hydraulics, control system interactions, reactor kinetics, and transport of noncondensable gases. Although previous versions of the code have included the analysis of fission product transport and deposition behavior using models derived from TRAP-MELT[4], this capability is being replaced through a data link to the detailed fission product code, VICTORIA, as a result of an effort to reduce duplicative model development and assessment. The SCDAP models calculate the heatup and damage progression in the core structures and the lower head of the reactor vessel. The calculations of damage progression include calculations of the meltdown of fuel rods and structures, the fragmentation of embrittled fuel rods, the formation of a molten pool of core material, and the slumping of molten material to the lower head.

SCDAP/RELAP5 is capable of modeling a wide range of system configurations from single pipes to different experimental facilities to full-scale reactor systems. The configurations can be modeled using an arbitrary number of fluid control volumes and connecting junctions, heat structures, core components, and system components. Flow areas, volumes, and flow resistances can vary with time through either user control or models that describe the changes in geometry associated with damage in the core. System structures can be modeled with RELAP5 heat structures, SCDAP core components, or SCDAP debris models. The RELAP5 heat structures are one-dimensional models with slab, cylindrical, or spherical geometries. The SCDAP core components include representative light water reactor (LWR) fuel rods, silver-indium-cadmium (Ag-In-Cd) and B_4C control rods and/or blades, electrically heated fuel rod simulators, and general structures. A two-dimensional, finite element model based upon the COUPLE[5] code may be used to calculate the heatup of debris and/or surrounding structures. This model takes into account the decay heat and internal energy of newly fallen or formed debris and then calculates the transport by conduction of this heat in the radial and axial directions to the wall structures and water surrounding the debris. Perhaps the most important use of this model is to calculate the heatup of the vessel wall so that the time at which the vessel may rupture can be determined. Other system components available to the user include pumps, valves, electric heaters, jet pumps, turbines, separators, and

a. The MOD3.1 designates a generic version of the code. Where needed, specific developmental version identification will be included with the code name. For example, SCDAP/RELAP5/MOD3[8x] would specify developmental Version 8x.

accumulators. Models to describe selected processes, such as reactor kinetics, control system response, and tracking noncondensable gases, can be invoked through user control.

One of the important aspects of maintaining a detailed code, such as SCDAP/RELAP5, is the incorporation of new information gained through severe accident experiments, analytic programs, and user experience. In many cases new models or code improvements are being developed by researchers external to the INEL. As a result, a set of programming guidelines have been developed to facilitate the incorporation of the improvements and to insure that the code still maintains the same functionality on a wide variety of computer platforms. This manual is intended to provide both internal and external developers with a resource to document our experience and understanding of the code.

2. CODE ARCHITECTURE

Modeling flexibility, user convenience, computer efficiency, and design for future growth were primary considerations in the development of SCDAP/RELAP5. The following sections describe computer adaptability, code organization, input processing, and transient operation.

2.1 Computer Adaptability

2.1.1 Source Coding

SCDAP/RELAP5 is written in FORTRAN 77. Compile time-options are provided to allow operation on 60-bit machines and 32-bit machines that have double-precision (64-bit), floating-point arithmetic. A common source is maintained for all computer versions. Reported errors are resolved on all computer versions derived from the common source.

SCDAP/RELAP5 minimizes the need for hardware specific coding through the use of generic functions, character variables and statements, and open statements. The bit handling functions used are from a Mil-spec standard and many computers have implemented that standard. Some machine dependent coding is needed to overcome deficiencies in the Fortran standard. For example, the standard does not allow specification of variable range and precision requirements. Unless modified, the Fortran for a 60-bit machine in single precision would use 60 bits while the 32-bit machine would use only 32 bits. Additional statements are needed to indicate double precision on the 32-bit machine. Some compilers have options for automatically converting to double precision, but that is nonstandard and not available on all systems. Nonstandard coding is needed in some subroutines to define the smallest and largest floating point numbers, the smallest floating point increment, and to access the radix, fraction, and power part of a floating point number. The next standard should remedy these, but until then, precompilers are used to handle hardware and software differences.

2.1.2 Systems

The SCDAP/RELAP5 computer program should execute on a wide variety of scientific computers with minimal modifications. In particular, the code should execute on all 60-bit computers, that is computers using 60 bits for both floating point and integer arithmetic. It should also execute on the multitude of 32-bit computers that range from workstations to supercomputers and that have 32-bit integer arithmetic but provide 64-bit floating point arithmetic through double precision operations. The code is maintained for all computers in a common source file, and through one or two stages of precompiling, the code is made suitable for a particular computer.

2.2 Organization of the Code

SCDAP/RELAP5 is coded in a modular fashion using top-down structuring. The various models and procedures are isolated in separate subroutines. Figure 1 shows an overview of the code architecture.

Input processing is performed in INPUTD and associated subroutines. Transient control is performed by TRNCTL and associated subroutines. The STRIPF routine extracts data from the restart plot file for use in other computer programs. Because of their complexity, the input processing and transient control

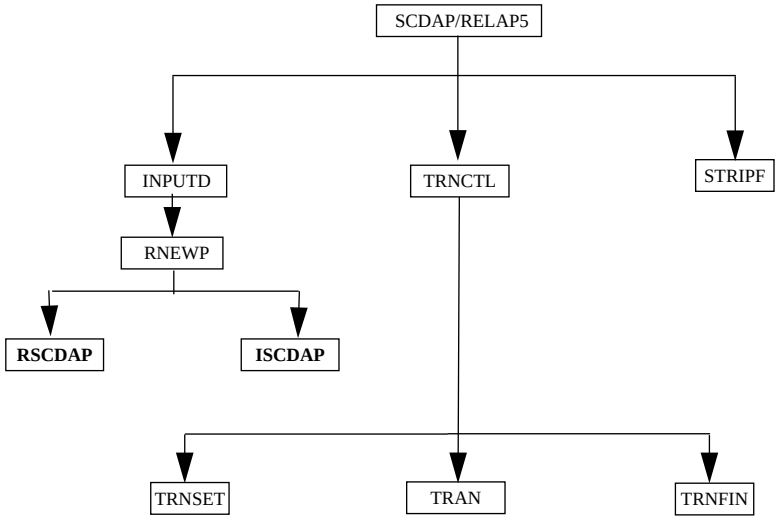

Figure 1. SCDAP/RELAP5 general architecture.

routines are described in more detail in Sections 2.2.1 and 2.2.2.

2.2.1 Input Processing Overview

The input processing is performed in three phases. In the first phase, the input data is read and checks are made for typing and punctuation errors (such as multiple decimal points and letters in numerical fields), and stores the data keyed by card number so that the data are easily retrieved. A listing of the input data is provided, and punctuation errors are noted.

During the second phase, restart data from a previous simulation are read if the problem is a RESTART type, and all input data are processed. Some processed input is stored in fixed common blocks, but the majority of the data are stored in dynamic data blocks that are created only if needed by a problem and sized to the particular problem. In a NEW-type problem, dynamic blocks must be created. In RESTART problems, dynamic blocks may be created, deleted, added to, partially deleted, or modified as modeling features and components within models are added, deleted, or modified. Extensive input checking is done, but at this level checking is limited to new data from the cards being processed. Relationships with other data cannot be checked because the latter may not yet be processed.

The third phase of processing begins after all input data have been processed. Because all data have been placed in common or dynamic data blocks during the second phase, complete checking of interrelationships can proceed. Examples of cross-checking are existence of hydrodynamic volumes referenced in junctions and boundary conditions; entry or existence of material property data; and validity of variables selected for minor edits, plotting, or used in trips and control systems. As the cross-checking proceeds, cross-linking of the data blocks is done so that it need not be repeated at every time step. The initialization required to prepare the model for start of transient advancement is done at this level. As errors are detected, various recovery procedures are used so that input processing can be continued and a maximum amount of diagnostic information can be furnished. Recovery procedures include supplying

default or replacement data, marking the data as erroneous so that other models do not attempt use of the data, or deleting the bad data.

As was shown in Figure 1, the first phase of input processing for the SCDAP portion of the code is performed in subroutine RSCDAP. Details of this subroutine are shown in Figure 2. These subroutines perform as follows. First, the severe core damage sections of the code are initialized with calls to SCDCON and PREINT. Second, the general bundle information is read within RSCDAP. This information includes general bundle geometry information, as well as grid spacer information. The third step is to read the component specific information. This is accomplished by calls to the component specific input routines, RFUEL (fuel rod), RCYLIN (Ag/In/Cd control rod), RSHROD (shroud), RBWR (B_4C control rod), RFUELE (fuel element), and RBLA (Control Blade/ Channel Box). If the component specific information is read successfully, RSCDAP then calls a component specific initialization routine. Finally, the radiation heat transfer information, such as view factors and path lengths are read in routine RRADIA.

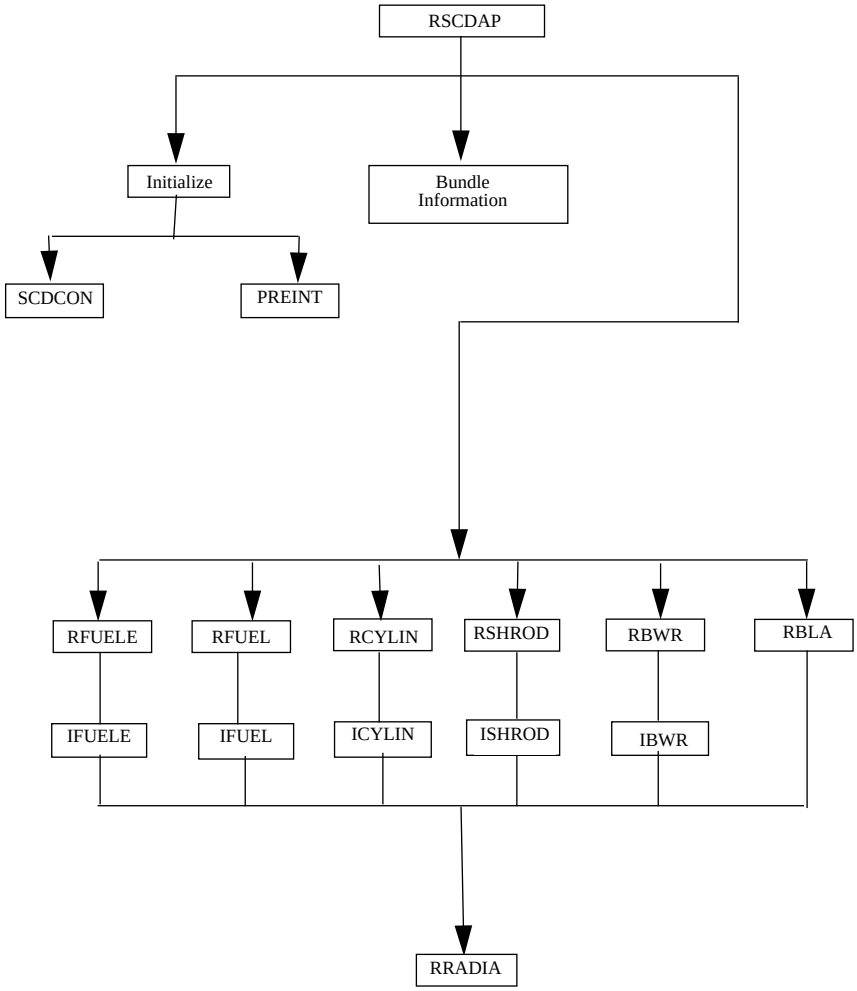

Figure 2. SCDAP/RELAP5 input architecture.

2.2.2 Transient Overview

Subroutine TRNCTL, shown in the center branch of Figure 1, consists only of the logic to call the next lower level routines. Subroutine TRNSET brings dynamic blocks required for transient execution from disk into computer central memory, performs final cross-linking of information between data blocks, sets up arrays to control the sparse matrix solution, establishes scratch work space, and returns unneeded computer memory. Subroutine TRAN, the driver, controls the transient advancement of the solution. Nearly all the execution time is spent in this subroutine, and TRAN is also the most demanding of memory. The subroutine TRNFIN releases space for the dynamic data blocks that are no longer needed and prints the transient timing summary.

Because of the complexity of the transient architecture, the transient overview will be divided into sections. Section 2.2.2.1 discusses the thermal advancement, Section 2.2.2.2 will discuss the severe accident model advancement, and Section 2.2.2.3 discusses the hydrodynamic advancement.

2.2.2.1 Thermal Advancement

The control of the time step advancement, the trip logic models, the calculation of the thermodynamic state of each hydrodynamic volume in the system, and the heat conduction solution for reactor heat structures are performed within this block of subroutines.

The following description is presented for selected transient subroutines (refer to Figure 3):

DTSTEP Determines the time step size, controls output editing, and determines whether transient advancements should be terminated. During program execution, this module displays such information as CPU time, problem time, and the maximum cladding temperature on a terminal screen.

TRIP Evaluates logical statements. Each trip statement is a simple logical statement which has a true or false result. The decision of what action is needed resides within the components in other modules. For example, valve components open or close the valve based on trip values; pump components test trip status to determine whether a pump electrical breaker has tripped.

TSTATE Calculates the thermodynamic state of the fluid in each hydrodynamic user-defined time-dependent volume.

HTADV Advances heat conduction/transfer solutions using previous-time-step reactor kinetics power and previous-time-step hydrodynamic conditions for computing heat transfer coefficients. It calculates heat transferred across solid boundaries of hydrodynamic volumes.

2.2.2.2 SCDAP Advancement

The advancement of the severe core damage models are controlled within the SCDAP advancement block. The entry point for this block of models is the SCDPRH subroutine. The SCDPRH subroutine advances the heat conduction solution (for core components only), the mechanical response models (including changes in geometry), and fission gas release models using previous-time-step hydrodynamic conditions. The SCDPRH routine drives four basic blocks of modeling, consisting of the heat transfer

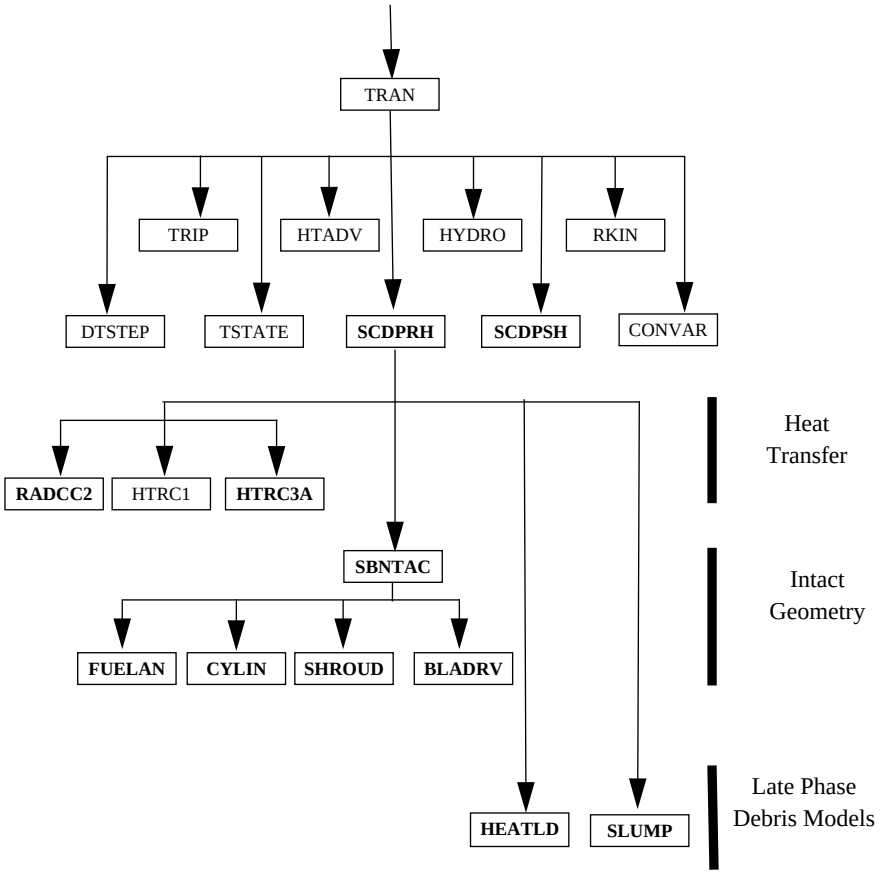

Figure 3. SCDAP/RELAP5 transient architecture.

block, the intact geometry models, the in-core debris models, and the core slumping model.

The first block of modeling driven by SCDPRH consists of calls to the heat transfer models within RADCC2, and HTRC1/HTRC3A. RADCC calculates the radiation heat transfer in a fuel bundle, and is only made difficult by the interface to the MINERVA mathematics library for the solution of the system of differential equations. HTRC1 computes heat transfer from the intact geometry routines to the coolant. HTRC1 computes heat transfer coefficients for air-water mixtures, single phase liquids, subcooled nucleate boiling, saturated nucleate boiling, subcooled transition film boiling, saturated transition film boiling, subcooled film boiling, saturated film boiling, and single phase vapor convection. HTRC3A calculates heat transfer from debris to coolant.

The second significant block of coding within SCDPRH is the call to SBNTAC, which drives all the intact geometry routines, including the rubble debris bed models. The core component routines calculate the response of include FUELAN (LWR fuel rod), CYLIN (Ag/In/Cd control rod), SHROUD (shroud), and BLADRV (control blade/channel box component).

The third significant block of coding is the call to HEATLD, the routine which controls the in-core debris bed models. This routine calculates the heatup of circulating liquefied debris contained by hardpan,

as well as the thickness of the hardpan and spreading of the molten pool.

Finally SCDPRH calls SLUMP, which controls the transfer of material to the COUPLE sub-code. This routine determines whether a new unique slumping of core material into the lower vessel region occurred during the time step. If slumping occurred, it calculates the total mass of material that will end up eventually falling into the lower vessel region due to this slumping. This falling may be spread out over many time steps, or occur instantaneously.

The advancement of the detailed COUPLE model for lower plenum debris heating and melting is controlled by SCDPSH, which is exercised after the hydrodynamic solution has been achieved. This block of routines will not be discussed since no external (non-INEL) model development activities are currently planned for these models.

2.2.2.3 Hydrodynamic Advancement

The top-level organization of SCDAP/RELAP5 allows advancement of the severe accident block prior to the hydrodynamic calculation, and allows the severe accident block to have significantly greater control of the decision as to whether a specified set of conditions are acceptable, thereby influencing the time step control.

HYDRO Advances the hydrodynamic solution.

RKIN Advances the reactor kinetics of the code. It computes the power behavior in a nuclear reactor using the space-independent or point kinetics approximation which assumes that power can be separated into space and time functions.

CONVAR Provides the capability of simulating control systems typically used in hydrodynamic systems. It consists of several types of control components. Each component defines a control variable as a specific function of time advanced quantities. The time advanced quantities include quantities from hydrodynamic volumes, junctions, pumps, valves, heat structures, reactor kinetics, trip quantities, and the control variables themselves. This permits control variables to be developed from components that perform simple, basic operations.

3. DATA BASE

Modeling flexibility, user convenience, computer efficiency, and design for future growth were primary considerations in the development of SCDAP/RELAP5 database. The following sections present information for the programming and implementing of new models. Lists of typical significant variables are listed in Section 3.1, as well as procedures to calculate the values of these variables as a function of space. Section 3.2 presents examples of how information is extracted from the SCDAP and RELAP5 data base for the purpose of defining initial conditions and boundary conditions for new models that may be implemented into the code. Section 3.3 describes methods to modify the data base, and Section 3.4 presents general requirements for programming.

3.1 Accessing The Existing Data Base

This section describes how programmers can access the existing database of SCDAP/RELAP5. The scheme for obtaining these variables as a function of space is first described. An identification is then made of the variables that are most likely to be used by new models. All significant parameters which are tracked within SCDAP/RELAP5 are included in common blocks. Each common block, and a listing and explanation of the variables within that common block are included in this report as Appendix B., "COMMON BLOCK DESCRIPTIONS". It is worth noting that all variables used in SCDAP/RELAP5 are in SI units. All data should be accessed directly from the common blocks. Although some parts of the code still pass information from one subroutine to the next through long argument lists, this practice has proven too difficult to maintain and has become obsolete.

Variables that vary with respect to location in the reactor core have two indexes to define the location at which the value of the variable is desired. The general form of the variables is v(j,k), where “v” is the name of a variable and "j" and "k" are indexes that define the location for which the value of the variable is wanted. The first index is the component number and the second index is the axial node number. The component number specifies the radial location within the reactor core and the axial node number specifies the elevation relative to the bottom of the reactor core. Figure 4 shows the relation of component number and axial node number to locations within the reactor core. In this figure, the reactor core has been divided into two radial segments and three axial nodes. The value of a fuel rod variable in the center radial segment is obtained by using a component index of one and the value of a fuel rod variable in the outer radial segment is obtained by using a component index of three. The value of a fuel rod variable at the bottom of the reactor core is obtained by using an axial node index of one. For example, the value of a fuel rod variable at the bottom center of the reactor core is v(1,1) and the value at the middle elevation and at the periphery part of the core is v(3,3).

Variables that vary within the domain of an individual component have three indexes. An example of a variable with three indexes is the variable tcond3(l,k,j), where "temp" is the temperature at a location defined by the indexes "j", “k”, and “l”. The first index is the radial node number within the component. The first radial node is at the centerline of the component and the largest radial node number is at the surface of the component. The last two indexes are the axial node number and component number, respectively. These last two indexes specify the location within the reactor core. A given component number can reside in any radial segment within the core. The radial segment within which it resides is defined by the code user.

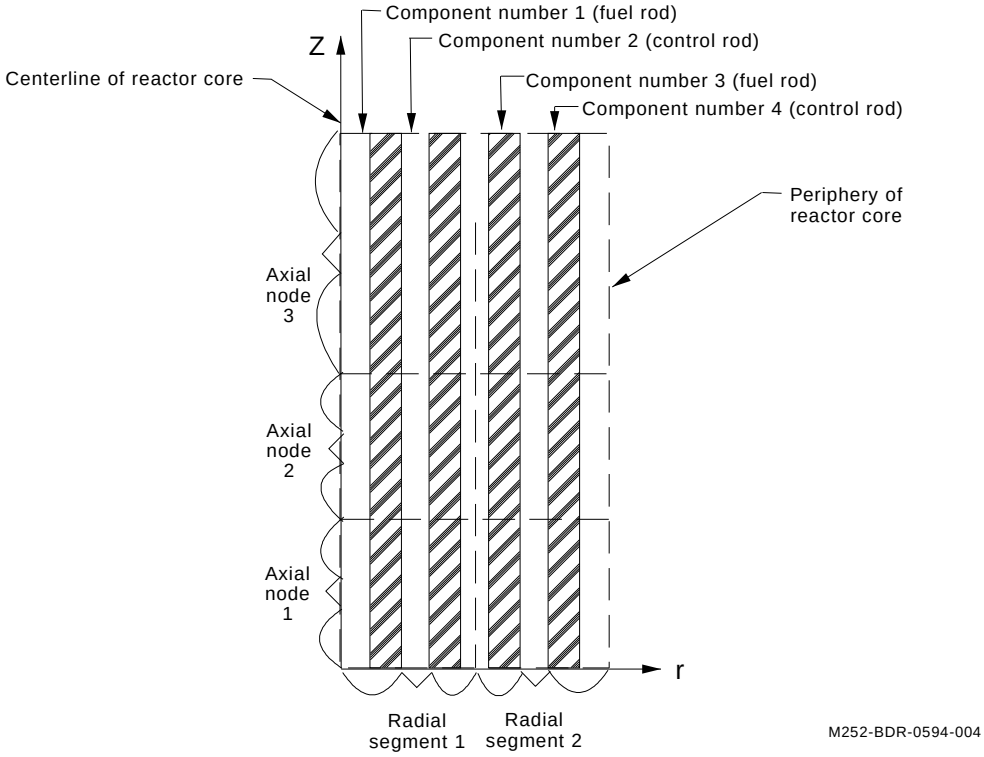

Figure 4. Scheme for defining location in reactor core.

Variables are stored that identify the components within each radial segment and the hydrodynamic control volume the represents the fluid within each radial segment. These variables are defined in Table 1. At a given elevation, one and only one hydrodynamic control volume represents the fluid within a given radial segment. Radiation enclosures may span several radial segments. Table 1 also lists the variables that define the relation of radiation enclosures to radial segments and component numbers.

Table 1. Variables To Compute Location Within Reactor Core

Fortran variable	Definition	Name of common block with variable
ncomp	Number of different components in reactor core	scddat
idcomp(j)	Identification of type of j-th component; 0 = fuel rod, 1 = PWR control rod, 8 = BWR control blade, 2 = shroud	ndxara
ndcomp	Maximum allowed number of components	scddat
naz(j)	Number of axial nodes in j-th component (all components have same number of axial nodes)	ndxara
ndax	Maximum allowed number of axial nodes	scddat

Table 1. Variables To Compute Location Within Reactor Core (Continued)

Fortran variable	Definition	Name of common block with variable
nvad((j-1)*ndax+k)	Offset index of RELAP5 control volume that interfaces with surface of jith component at k-th axial node; the RELAP5 variable filndx(4) is added to offset index to give absolute value of index	tblsp
nrods2(k,j)	Number of radial nodes at k-th axial node of j-th component	ndxara
zcond2(k,j)	Height above bottom of core of j-th component at axial node k (m)	scdout
dzcnd2(k,j)	Height of axial node k of component j (m)	scdout
xcond3(l,k,j)	Radius of l-th radial node at k-th axial node of j-th component (m)	scdcom
nsigl(j)	Number of individual rods in component j	ndxara
pitch(j)	Pitch (m) [pitch(j)*pitch(j) = cross section area of component and flow area associated with it]	scdout
igp	Number of radial segments	tblsp
icp(i)	Number of components in i-th radial segment	tblsp
ncmpgp(j)	Number of radial segment that j-th component resides in	tblsp
ngpc(i,jj)	Component number of the jj-th local component within radial segment i (jj has integer values ranging from 1 to ncmpgp(j))	tblsp
ngpv(i,k)	Index of RELAP5 control volume in i-th radial segment at k-th axial node	tblsp
numenc	Number of radiation enclosures	radata
ncpenc(n)	Number of radial segments within radiation enclosure number n	radata
nrepfc(n,ii)	Number of radial segment for ii-th local radial segment within n-th radiation enclosure (ii has integer values ranging from 1 to ncpenc(n))	radata
ishrd(n)	Number of component that encloses the n-th radiation enclosure (idcomp(ishrd(n)) must equal 2 and thus be a shroud type of component)	trnot1
ncompe(n)	Number of components in n-th radiation enclosure	radata

Table 1. Variables To Compute Location Within Reactor Core (Continued)

Fortran variable	Definition	Name of common block with variable
ngroup(n,jj)	identification number of the jj-th local component in the n-th radiation enclosure (jj has integer values ranging from 1 to ncompe(n))	radata
fviewg(n,j1,j2)	View factor from j1-th local component within n-th radiation enclosure to j2-th local component within this enclosure (j1 and j2 have integer values ranging from 1 to ncompe(n))	radata

The variables that store the calculated state of each location in the reactor core are described next. These variables are shown in Table 2. The information contained in these variables includes; 1. temperature, 2. state of degradation, 3. nuclear heat generation rate, 4. oxidation heat generation rate, and 5. material composition. In Table 2, the index “j” is component number, the index “k” is the axial node number, the index “l” is the radial node number, and “i” is the radial segment number. A location in the reactor core maintains its original indexes after that location has degenerated from intact rods to debris.

Table 2. Variables That Define State Of Reactor Core

Fortran variable	Definition	Common block with variable
tcond3(l,k,j)	Temperature at l-th radial node of k-th axial node of j-th component (K)	scdout
irubpp(k,j)	Indicator of whether degradation into porous debris has occurred; 0 = no, 1 = yes	tblsp
lcrucb(k,j)	Indicator of whether ceramic melting has occurred at location; 0 = no, 1 = yes, 2 = material at this location has slumped to lower head.	tblsp
damlev(k,j)	Indicator of level of damage; 0.0 = no damage, 0.1 = cladding ruptured, 0.2 = degeneration into porous debris, 1.0 = ceramic melting.	tblsp
nplblk(i,k)	Indicator of plane blockage; 0 = no, 1 = yes	tblsp
nbublk(i,k)	Indicator of bulk blockage; 0 = no, 1 = yes	tblsp
dzfrcq(k,j)	Fraction of height of node that is filled with relocated material.	tblsp
unuca(k,j)	Nuclear heat generation per unit length per individual rod (W/m)	scdcom
oxdhcc(k,j)	Oxidation heat generation per unit length per individual rod (W/m)	intcom

The variables that define the locations of the core that have degenerated into porous debris are shown in Table 3. This table also lists the variables that define the state of the debris, its internal heat generation rate, and its material composition. In this table, the index “n” references the n-th debris region in the core and the index “m” references the m-th axial node within the n-th debris region. The index “m” runs from a value of one for the bottom most axial node in the debris region to a value of (ndebup(n)-ndebbt(n)+1) for the top most axial node. Each debris region consists of one and only one fuel rod component. It includes PWR control rods embedded within the fuel rod component. The index “k” refers to the axial node in the core-wide framework and the index “j” refers to the component number within the core-wide framework.

Table 3. Variables That Define Location And State Of Debris

Fortran variable	Definition	Common block with variable
ndbreg	Number of debris region in reactor core	debcom
idbcom(n)	Number of component in n-th debris region, where n has integer values ranging from 1 to ndbreg	debcom
ndebbt(n)	Number of axial node at bottom of n-th debris region	debcom
ndebup(n)	Number of axial node at top of n-th debris region	debcom
porvol(m,n)	Porosity	debcom
ddbvol(m,n)	Particle diameter (m)	debcom
tmpdeb(m,n)	Temperature (K)	debcom
powdbl(m,n)	Heat generation rate (W/m^3)	debcom
voldeb(m,n)	Volume of debris per unit volume (m^3/m^3)	debcom
aovrdb(m,n)	Surface area of debris per unit volume (m^2/m^3)	debcom
qnchdb(m,n)	Heat transfer from debris to coolant (W)	debcom
wuorub(k,j)	Mass of UO^2 per rod at axial node k of component j (kg) [variable is not calculated until location has degenerated into debris]	tblsp
wzorub(k,j)	Mass of ZrO2 per rod at axial node k of component j (kg) [variable is not calculated until location has degenerated into debris]	tblsp
wssrub(k,j)	Mass of stainless steel per rod at axial node k of component j (kg) [variable is not calculated until location has degenerated into debris]	tblsp

Common block ‘voldat’ contains most of the thermal hydraulic variables involved in interfacing to severe accident models. SCDAP/RELAP5 uses mnemonics that relate variable name to variable definition, and the index for referencing thermal hydraulic data for axial node of each component is stored during input processing. A list of typical variables which may be of interest to code developers is included in

Table 4.

Table 4. Typical Thermodynamic Variables Of Interest To Model Developers

Variable	Units	Description
q(l)[a]	W	total rate of heat transfer from surface to fluid
qwg(l)	W	rate of heat transfer from surface to vapor
gammaw(l)	kg/m^3	volumetric vapor generation rate at surface
velf(k)	m/s	average velocity of liquid phase in volume with index k
velg(k)	m/s	average velocity of vapor phase in volume with index k
voidg(k)		vapor void fraction
rhof(k)	kg/m^3	density of liquid phase
rhog(k)	kg/m^3	density of vapor phase
satt(k)	K	saturation temperature
tempg(k)	K	temperature of vapor phase
viscg(k)	kg/m*s	viscosity of vapor phase
p(k)	Pa	average pressure within hydrodynamic volume
sathg(k)	J/kg	enthalpy of vapor at saturation conditions

a. l = nvad(axial node,fuel element number) + filndx(4)

3.2 Examples of Variable Use for Model Interface

This section presents examples of the use of variables in the SCDAP/RELAP5 data base for providing the initial conditions and boundary conditions that may be required by new models implemented into the code.

The first example shows how the fluid conditions at any location in the reactor core are obtained. The Fortran programming for obtaining the fluid conditions is shown below. Comments are added to explain the operations.

Find fluid conditions at the location in the reactor core of radial segment i at axial node k:

Let l = absolute value of SCDAP/RELAP5 volume index for this location.

l = ngpv(i,k)+filndx(4)
(filndx(4) is defined by SCDAP/RELAP5 and stored in the common block named comctl).

The SCDAP/RELAP5 arrays that store fluid conditions are in the common block named voldat.

The substitution of the index “l” into SCDAP/RELAP5 arrays results in values of the following con-

ditions:

p(l) = pressure of fluid at location of (i,k) (Pa),

tempg(l) = temperature of vapor phase of water at location of (i,k) (K),

voidf(l) = volume fraction of liquid phase of water at location of (i,k),

avol(l) = flow area at location of (i,k) (m^2),

diamv(l) = hydraulic diameter at location of (i,k) (m).

The second example shows how to find the fluid conditions at the surface of component number j at axial node k.

First find index of SCDAP/RELAP5 control volume that represents fluid at this location; name this index “l”.

l = nvad((j-1)*ndax + k) + filndx(4)

The substitution of the index “l” into SCDAP/RELAP5 arrays results in the following conditions:

p(l) = pressure of fluid at location of (j,k) (Pa),

tempg(l) = temperature of vapor phase of water at location of (j,k) (K),

velg(l) = velocity of vapor phase of water at location of (j,k) (m/s),

volno(2,l) = 9-digit volume number of SCDAP/RELAP5 control volume at location of (j,k).

The third example shows how information is extracted from the SCDAP/RELAP5 data base and used by a debris model. It is assumed that the information is wanted for the second debris region in the reactor core.

First, define index “n” to apply to second debris region.

n= 2

The fuel rod component that is in this debris region is idbcom(n).

The radial segment of the core that this debris region is located in is ncmpgp(idbcom(n)).

The elevation of the bottom of this debris region is equal to zcond2(ndebbt(n),idbcom(n)) - 0.5*dzcnd2(ndebbt(n),idbcom(n)).

The elevation of the top of this debris region is equal to zcond2(ndebtp(n),idbcom(n)) + 0.5*dzcnd2(ndebtp(n),idbcom(n)).

The temperature of the debris at the bottom axial node in the debris region is equal to tmpdeb(1,n).

The volumetric heat generation rate at the bottom axial node in the debris region is powdbl(1,n).

The porosity of the debris at the bottom axial node is porvol(1,n).

The index of the SCDAP/RELAP5 control volume that represents the fluid in the bottom axial node in the debris region is calculated by the operation:

```
l = ngpv(ncmpgp(idbcom(n)),ndebbt(n)) + filndx(4)
```

The fluid conditions at this location are then determined by substituting the index “l” into the SCDAP/RELAP5 array that store fluid conditions. If additional fluid conditions are required, the developer should refer to Appendix D., "THERMODYNAMIC PROPERTY PACKAGE".

If the debris model calculates that a blockage to flow in the axial direction occurs in the second debris region at its bottom axial node, then the following operations are performed to impose the blockage in the thermal hydraulic model:

```
nplblk(ncmpgp(idbcom(n)),ndebbt(n)) = 1
lmap = invfnd(l)
do 100 jn=1,invcnt(lmap)
ja = invvnx(lmap)
if(iand(jc(ja),16384).eq.0 .and. ij1nx(ja).eq.1)go to 130
lmap = lmap +invskp
100 continue
130 continue
```

An example of imposing a plane blockage is shown in the SCDAP/RELAP5 subroutine named scdad4. The operations shown above use several hydrodynamic variables, such as invcnt(lmap). These variables are stored in the common block named invtbl. The problem time and hydrodynamic time step are variables that are used by many models. The problem time is stored in the variable “timehy” in the common block named contrl. The current hydrodynamic time step is stored in the variable “dt” in this same common block.

3.3 Maintaining the Data Base

3.3.1 Adding Parameters

To avoid building unnecessary limits into the models, modelers must avoid the use of fixed dimensions and do loop indices. Instead use the parameters which have been defined in common block ‘scddat’. The parameters ndcomp, which is the maximum number of components, ndax, the maximum number of axial nodes, and ndrd, the maximum number of radial nodes, are particularly useful to modelers working within the intact geometry models.

To assure that the names of variables in common blocks do not conflict with one another, all RELAP5 common blocks are included in routine ‘RELAP5’, and all severe accident common blocks are included in routine ‘scdadv’. New common blocks should also be included in ‘scdadv’, even if none of the data within the common block is used within the routine.

All common blocks are to have an associated block of comment statements, which explain the func-

tion and limits of each variable in the common block, much as is documented in Appendix B. As an example, each of the variables in common block ‘scddat’, have been documented in common block ‘scddatc’. This practice allows code maintenance staff and other developers to include extensive variable documentation in routines where needed, while eliminating extensive comment cards from each routine that includes the common block.

3.3.2 Restart Capability

One of the requirements for new models is that they must have restart capability. This capability allows the user to repeat or restart the calculation at a predetermined point. This capability already exists in the code, and the programmer need only specify which common blocks are to be written to the restart/plot file. The programmer is strongly encouraged to identify only those common blocks which need to be saved for restart capability. The size of the restart file can increase dramatically with each additional common block saved.

The identification of common blocks to be written to the restart file is done in subroutine ‘GNINIT’. First the index i is incremented, then the common block identifier is saved as array element comnam(i). Thirdly, the memory location of the first variable in the common block is stored in array element comdat(i), and the memory location of the final variable, which is always an integer variable, mcmeXX (where XX is a unique identifier) is stored in array element comdln(i). Finally a flag is set by setting array element filflg(i) equal to 4. Because of this procedure it is important that the programmer NOT resequence either the first or last element of each common block. In this case the restart capability of the code would be compromised. Adding variables to existing common blocks only requires that the alignment of the variables be correct (See “Common Block Guidelines” on page 20.), and that the first and last variables in the common block remain unchanged.

One of the acceptance tests for each new model, prior to it’s incorporation into the code, will be the ability to restart in the middle of a transient, with no change in results.

3.3.3 Time Step Repetition Capability

One of the new capabilities that has been added to SCDAP/RELAP5 is the ability to repeat a time advancement at a reduced time step under certain conditions. In order to maintain this capability, new models must be able to capable of resetting all significant variables to their beginning-of-time-step value, utilizing the time step repetition capability.

The minimum ability to repeat a time advancement at a reduced time step can be implemented in four steps. The first step is separate the data base into two types of variables, primary and secondary. Primary variables are those variables which are not derived from other variables, which secondary variables are the derived variables. The second step is to define parallel arrays for each of the primary variables. These parallel arrays will hold the beginning-of-time-step values. The third and fourth steps are to modify subroutine ‘SCDMOV’. This routine is called at the end of each time advancement, successful or unsuccessful. The first half of this routine is executed if the time advancement was unsuccessful (succes .ne. 0) and performs the function of moving the beginning-of-time-step values into the working arrays. The second half of subroutine ‘SCDMOV’ is executed if the time advancement was successful, and moves the end-of-time-step

values into the beginning-of-time-step arrays.

The programmer for all new models should consider whether some aspect of their model should be included in the success criteria for a time advancement. If a calculation can be driven non-physical because of a too-large time advancement or if significant amounts of ‘stress’ are being placed on the thermodynamic solution, then that model should be included in the success criteria for a time advancement.

3.4 General Requirements

- All data should be accessed directly from the common blocks. Accessing data by passing it on call statements is not acceptable.
- New models must have restart capability.
- Programmers should not resequence either the first or last variable of each common block saved to the restart file. This will seriously compromise the restart capability.
- Identify only the common blocks which are truly needed for restart capability. Unneeded common block storage can waste significant amounts of file space for each restart file.
- One of the acceptance tests for each new model, prior to it’s incorporation into the code, will be the ability to restart in the middle of a transient, with no change in results.
- New models must be able to utilize the time step repetition capability.
- Use predefined parameters as array dimensions and/or do loop indices, rather than ‘hardwiring’ values into the models.
- All common blocks should be called in ‘scdadv’. Each variable within a common block should be documented in a parallel “comment” common block.

4. COMPUTER COMPATIBILITY

This section lists some of the programming rules to allow the SCDAP/ RELAP5 program to execute on a wide variety of computers. Preservation of the ability to run on both 64 bit machines such as Cray and 32 bit machines such as workstations does not require great effort but it does require some consistency.

4.1 Precompiler Utilities

The SELECT utility is applied to source code for all machines, and is intended to activate or deactivate specific sections of coding which are machine or option specific. SELECT first examines the list of variables which have been defined at the head of each file. It then parses each line of the source, to determine if the line should be left activated or commented out. As an example, consider the following:

```
$define option1
*if,def,option1
      LINE 1
*endif
*if,def,option2
      LINE 2
*endif
*if,-def,option2
      LINE 3
*endif
```

The SELECT utility will detect the definition of ‘option1’, leave LINE1 activated, comment out LINE2, and leave LINE3 activated. The utility will then convert each ‘*call comdeck’ statement to an include statement. The include statements are allowed in most Fortran compilers and the comdecks are copied into the source during compilation. The update program if used would have already applied the *if def logic and would have replaced *call statements with the comdecks.

The CNV32 program applies additional logic prior to compilation to allow operation of the program on 32 bit machines. On 32 bit machines, floating point operations are performed as 64 bit operations using double precision floating point operations. To automate this change to double precision, CNV32 changes all real statements without an asterisk immediately following to real*8. Also all floating point literals are changed to double precision. Thus 2.05 becomes 2.05d0 and 2.65e8 becomes 2.65d8.

The 32 bit machines allow only 32 bit operations. This causes difficulties with the integer and floating point equivalencing which is used to gain dynamic storage assignment with good mnemonics but without long argument lists. Most subscripted integer or logical variables equivalenced to floating point variables will probably need to appear as 64 bit quantities. This is done by adding a subscript to the integer and logical variables. A variable such as ia(5) is changed to ia(l,5) in equivalence statements and ia(2,5) in all other statements. No change is made if the reference to ia does not have subscripts. A list of global variables needing this change is maintained in the file ‘mlist’ and is read by CNV32. Local variables needing this change are indicated by a series *in32 variable_name lines terminated by an *in32end line.

Do not use format information imbedded in the read or write statement if it includes ‘e’ type format specifications. The CNV32 program does not scan for literal in format statements but it gets confused between literals with ‘e’ and ‘e’ formats that can appear in read and write statements.

4.2 Integer Array Equivalence to Floating Point in Argument List

Non-standard FORTRAN notation has been implemented to allow the pre-compiler, CNV32, to handle processing integer arrays in subroutine arguments. The use of the left square bracket ‘[’, in the integer

array specification forces the precompiler to treat this argument as an integer array stored within a bulk storage array that is a real array. As an example, if the programmer wished to pass an integer array element into a subroutine, and equivalence the element in the called routine to a real variable, then the correct method of programming would be:

```
$if -def,in32,1
      call polat(gtinfo(i),gtbl(i),.....)
$if def,in32,1
      call polat(gtinfo[i), gtbl(i),.....)
```

In the call to POLAT, the array gtinfo stores information for a RELAP5 general table within the bulk storage array in RELAP5 named “a”, which contains a mixture of real and integer variables. The line “$if def,in32,1” instructs the precompiler to treat the next line of coding in a special manner for 32 bit computers. The left square bracket in the argument gtinfo[i) instructs the precompiler to treat this argument as an integer array stored within a bulk storage array that is a real array.

4.3 Implicit Typing

The main program and most subprograms are compiled with implicit none. To allow other subprograms to be converted to implicit none, all comdecks containing common block or dynamic storage, definitions must have all variables typed. That is, all variables must appear in integer, real, logical, or character statements. This has been done for the current comdecks. As new variables are added to the common blocks, they must be typed by the appropriate type statement. This must also be done on new comdecks.

Every subprogram must have either (but not both)

```
$if def,in32,1
      implicit real*8(a-h, o-z)
```

or

```
$if def,impnon,1
      implicit none
```

The latter is preferable. The global variables are already typed.

The Cray compiler requires that all subprograms (routines called by a call statement) and all non-intrinsic functions must appear in an external statement. This is therefore a recommendation for all developers.

One of the peculiarities of programming across multiple platforms is the need to provide re-entrant coding. One of the requirements of re-entrant coding is the need to recompute local variables upon each call. There are machines which use dynamic variable allocation, which therefore do not retain the value of a local variable when the subroutine is exited. A second consideration, is the ability of SCDAP/RELAP5 to repeat a time advancement. In either case, the need exists to recompute local variables upon each call to the subroutine.

4.4 Common Block Guidelines

Common blocks must be organized to avoid alignment errors. An alignment error occurs when a double precision floating point number does not lie on a eight byte boundary (byte address divisible by 8). Thus the statement

common /any/ a,iz,b

would have an alignment error but

common /any/ a,ix,iz,b

would not. A good practice is to order common variables such that all real variables are first followed by integer and logical variables.

Common blocks containing floating, integer, or logical data cannot also contain character data. Character data should be typed character whenever possible. However, character input data is returned by input routines such as inp2 as real (64 bit machines) or real*8 (32 bit machines). Internal write statements should be used to convert these to character data for processing. If character data must be put into dynamic storage or common blocks written to the restart file, the character data should be converted to real by internal write statements.

4.5 General Requirements

The following requirements should be followed to insure compatibility with the existing code structure.

- Do not use implicit typing. Explicitly type each variable, and declare each external.
- Use generic function names. Thus use log instead of alog, and min instead of min0 or amin1. Most Fortran manuals have tables giving both functions and their generic names.
- Do not use tab characters for spacing. Many compilers generate error messages when tabs are encountered, and the INEL utilities cannot handle them.
- Use ‘e’ rather than ‘d’ to indicate floating point constants that need specification of a power of ten. The CNV32 will convert them to the ‘d’ notation, if necessary.
- Do not use format information imbedded in the read or write statement if it includes ‘e’ type format specifications. The CNV32 program does not scan for literal in format statements but it gets confused between literals with ‘e’ and ‘e’ formats that can appear in read and write statements.
- Local variables must be recomputed for each call to a subroutine. Dynamic allocation, and time step repetition may make the local variable invalid.

5. INPUT AND OUTPUT REQUIREMENTS

5.1 Input

The following sections describe the general methodology of getting data into SCDAP/RELAP5, what kind of checking to perform, and provides some general input guidelines.

5.1.1 Use Of The INP Package

The INP set of subroutines, documented in Appendix C, constitutes a convenient data input package for use with Fortran programs. The INP package is recommended for use since it offers several advantages to both program user and programmer. To the user, the package offers: free form input; card number to identify the cards; automatic removal of cards containing duplicate card numbers; arbitrary ordering of input cards except for duplicate cards; arbitrary use of comment cards and comments on data cards; ease of preparing cases in which only moderate changes are made from case to case; and a listing of the card data. For the programmer, the package is a convenient method for implementing a highly user oriented data input scheme and includes: extensive checking of the amount and mode (integer, floating point, or alphanumeric) of data; automatic expansion of sequential and overlay type of input data; deletion of unneeded cards; checking whether extraneous input has been entered; and the ability to detect several errors during input checking. The INP package also uses an output subroutine which reduces the programmer burden in formatting output pages.

The INP package forms a sorted table of card numbers cross-referenced to a list. The list contains data words obtained from the cards and mode words generated during input conversion. Because of the sorted table, the order of cards in the data deck is not important. The subroutine INPLNK accesses the table and can locate one card at a time. The subroutine INPMOD is used to check the appropriate mode of the data against a specified list, also one card at a time. The INP2 subroutine is used to check data and to move data from the list to a specified array. It can be used to process data from a single card or from a set of cards numbered within a specified range, and to check for minimum and maximum numbers of items and appropriate mode.

The CVI subroutine, documented in Appendix A, is called from the INP package. Although not routinely used by SCDAP/RELAP5 code developers, it has been included in this document because it is used by the code installation procedure, in the generation of the steam tables.

5.1.2 Input Checking

Input checking should be performed in three stages. The first stage, checking for the expected amount of data and the correct mode, is performed within the INP structure itself. The final two stages are to be performed by the input routine itself. These stages are, first, range checking and, second, error checking. Range checking consists of checks for range of normal use. Messages regarding out of range conditions should be warning messages (leading with 8 ‘$’) and should not be considered serious enough to cause termination after input processing. Error checking consists of checking for input which is not physical or which cannot be meaningfully applied. Messages for error checking should be fatal messages (leading with 8 ‘*’). The logical variable ‘FAIL’ should be set to true and the input parameter should be reset to a benign value to permit continued input processing. Since ‘FAIL’ has been set true, the case will be terminated after input processing.

Although SCDAP/RELAP5 uses SI units internally, the capability does exist to allow the user to use either British or SI units on input. This is accomplished by checking the state of a logical variable, ‘uniti’, in common block ‘contrl’. If uniti is set to true, then the user has specified that SI units are to be input, and

if false, then British units are to be input.

5.1.3 General Input Guidelines

The input philosophy of SCDAP/RELAP5 is to avoid the use of user options in both model selection and parameters. It is the policy of the INEL staff that, if the developer is unable to define a parameter, it unfair to expect a code user to do so, unless the physics of the model are so little understood, that the user must perform a parametric study. Where necessary, input parameters can be tolerated if a default can be defined.

5.2 Output

SCDAP/RELAP5 provides the code user with four types of output to observe the effects of new models, major edits, minor edits, plot variables, and real time messages. The following sections will describe how the programmer can access these capabilities.

5.2.1 Major Edit

Major edits are intended to provide a snapshot of plant conditions at a user selected frequency. They are tabular in nature and an effort has been made to group similar output. The subroutine which drives major edit output is ‘MAJOUT’, which in turn drives ‘MAJSCD’ which generates the severe accident output.

The programmer should, once again, allow the capability of generating major edit output in either british or Si units, depending on the state of the logical variable, ‘unito’. If ‘unito’ is true, SI output units have been requested, and if false, british units have been requested.

5.2.2 Minor Edit / Plot Capability

The capability of using either minor edit or plot variables is handled within the routine ‘SCNREQ’. Parameter’s desired for minor editor or plot output should be placed in a common block callable from ‘SCNREQ‘. Assistance can then be requested from the INEL staff in modifying ‘SCNREQ’.

5.2.3 Real Time Messages

The use of real time messages should be avoided since it is easy for the code user to miss these messages. Where necessary, a real time message should contain, as a minimum, the name of the generating routine, the time of the event, and a brief message describing the event. The programmer should be aware that a significant number of messages may be generated, since the frequency is set by the transient conditions rather than the user.

5.3 Requirements

5.3.1 Input

- The use of the INP routines for input is mandatory.
- Allow use of British or SI units on input.
- Perform error and range checking. Allow processing to continue with benign values.
- Indiscriminate use of user options is not acceptable.

- Gather input in a logical manner. That is, group related input on the same input card.

5.3.2 Output

- Gather output in a concise and clear manner.
- Allow use of British or SI units on output.
- Use of real time messages is discouraged. If necessary, identify subroutine issuing message.
- Identify significant parameters for major edit printout.
- Identify less significant parameters for minor edit or plot capability.

6. BIT PATTERNS

Bit patterns are used throughout SCDAP/RELAP5 because they are a compact method of transferring large amounts of binary data, such as whether a specific model is to be used or not, between diverse modules.

6.1 Use of Bit Patterns

The use of bit maps is relatively easy for the programmer. Functions ‘iand’ and ‘ior’ are used to set or read each bit. As an example, assuming that the programmer wished to ‘turn on’ the SCDAP restart bit (bit 7 of SCNTRL), then the following FORTRAN statement would be used.

scntrl = ior(scntrl,64)

If the programmer then wished to test the same bit, to determine it’s state, then the following FORTRAN statement would be used to see if the bit was ‘off’,

if(iand(scntrl,64).eq.0) ...

and the following FORTRAN statement would be used to see if the bit was ‘on’

if(iand(scntrl,64).eq.1) ...

In a similar manner, multiple bits may be tested at the same time. As an example, if the programmer wished to test for bit position’s 1 and 4 both being on, the following statement would be used

if(iand(scntrl,9).eq.9) ...

6.2 Severe Accident Control

The bit pattern to examine severe accident model controls are in the SCDDAT common block variable SCNTRL.

Table 5: SCDAP Control Bit Pattern (SCNTRL)

Position	Value	If Bit Is ’on’, Model Activated
1	1	SCDAP
2	2	New Style SCDAP Input
3	4	Old Style SCDAP Input
4	8	COUPLE
5	16	New Style COUPLE Input
6	32	Old Style COUPLE Input
7	64	Restart
8-32		Currently Unused

6.3 Hydrodynamic Junction Control

There are two bit maps for hydrodynamic junction control, ‘jc’ and ‘jcex’, both in common block ’JUNDAT’.

Table 6: Junction Control Bit Pattern (JC)

Position	Value	Explanation
1	1	Choking Flag
2	2	Time Dependent Junction
3	4	Reversed “From” Volume Connection
4	8	Reversed “To” Volume Connection
5	16	No Choking
6	32	Old Time Choking
7	64	Choking Test For Accumulator
8	128	Input Flag
9	256	Abrupt Area Change
10	512	Two Velocity - One Velocity
11	1024	Separator
12	2048	Stratified Flow
13	4096	From Cross Flow Junction Option
14	8192	To Cross Flow Junction Option
15	16384	Cross Flow Flag
16	32768	Accumulator Active
17	65536	Horizontal Stratification Flag
18-19	2^{17}-2^{18}	Stratification Input Data
20-22	2^{19}-2^{21}	Jet Mixer Flags
23-25	2^{22}-2^{24}	Separator Flags
26	2^{25}	Unused
27	2^{26}	Horiz - Vert Justification
28	2^{27}	Up - Down Justification
29	2^{28}	Valve Flag

Table 6: Junction Control Bit Pattern (JC) (Continued)

Position	Value	Explanation
30	2^{29}	2nd Turbine Junction Flag
31	2^{30}	Unused
32	2^{31}	Unused

The extended junction control word was added to provide additional binary information. This word is documented in Table 7, “Extended Junction Control Bit Pattern (jcex)“.

Table 7: Extended Junction Control Bit Pattern (jcex)

Position	Value	Explanation
1		unused
2		ccfl flag
3		input ccfl flag
4-9		junction flow regime number
10-12		to face-1 bits
13-15		from face-1
16		input donor pressure flag
17		water packer junction flag

6.4 Hydrodynamic Volume Control

A word to hold binary data for the hydrodynamic volumes has been developed, comparable to the junction bit patterns described in the previous section. This word is documented in Table 8, “Volume Control Bit Pattern (vctrl)“.

Table 8: Volume Control Bit Pattern (vctrl)

Position		Explanation
1		time dependent volume flag
2		equilibrium flag
3		wall friction flag
4		input flag
5		vapor disappearance flag
6		accumulator flag

Table 8: Volume Control Bit Pattern (vctrl) (Continued)

Position		Explanation
7		pump flag
8		input water packer flag
9-19		new status flags, initialization type during input
20-30		old status flags
31		input bundle flag

The flow regime information for the hydrodynamic system is carried in bit pattern 'imap'. This word is documented in Table 9, "Flow Regime Bit Pattern (imap)".

Table 9: Flow Regime Bit Pattern (imap)

1-6		Flow regime map information
7		unused
8		horizontal stratification flag
9		unused
10		input vertical stratification flag
11-12		vertical stratification flag
13		experimental friction being used
14		wall friction input flag
15		flag that coordinate direction is being used
16		water packer input flag
17		bundle input flag
18		volume in multid component
19-24		flow regime number
25-26		metal appearance flags
27		laminar friction factor, 64 if 0, 96 if 1
28		ans input flag
29		level tracking flag.
30		reflood flag

Table 9: Flow Regime Bit Pattern (imap (Continued))

31		water packer volume flag

6.5 General Requirements

The following requirements should be used.

- Bit map patterns should be used for high level logic or control functions to minimize the transfer of large amounts of binary data such as turning models off or on.
- Existing bit patterns should be used where appropriate and cannot be altered without INEL concurrence.

7. CONCLUSIONS

One of the important aspects of maintaining a detailed code, such as SCDAP/RELAP5, is the incorporation of new information gained through severe accident experiments, analytic programs, and user experience. In many cases new models or code improvements are being developed by researchers external to the INEL. As a result, a set of programming guidelines have been developed to facilitate the incorporation of the improvements and to insure that the code still maintains the same functionality on a wide variety of computer platforms. This manual provides both internal and external developers with a resource to document our experience and understanding of the code. A number of general requirements for new models have been specified in each section. The following summarizes these requirements. Refer to the specific section for a more detail discussion.

These requirements are from the section entitled 'COMPUTER COMPATIBILITY'.

- Do not use implicit typing. Explicitly type each variable, and declare each external.
- Use generic function names. Thus use log instead of alog, and min instead of min0 or amin1. Most Fortran manuals have tables giving both functions and their generic names.
- Do not use tab characters for spacing. Many compilers generate error messages when tabs are encountered, and the INEL utilities cannot handle them.
- Use 'e' rather than 'd' to indicate floating point constants that need specification of a power of ten. The CNV32 will convert them to the 'd' notation, if necessary.
- Do not use format information imbedded in the read or write statement if it includes 'e' type format specifications. The CNV32 program does not scan for literal in format statements but it gets confused between literals with 'e' and 'e' formats that can appear in read and write statements.
- Local variables must be recomputed for each call to a subroutine. Dynamic allocation, and time step repetition may make the local variable invalid.

These requirements are from the section entitled 'INPUT AND OUTPUT REQUIREMENTS'.

- The use of the INP routines for input is mandatory.
- Allow use of British or SI units on input.
- Perform error and range checking. Allow processing to continue with benign values.
- Indiscriminate use of user options is not acceptable.
- Gather input in a logical manner. That is, group related input on the same input card.
- Gather output in a concise and clear manner.
- Allow use of British or SI units on output.
- Use of real time messages is discouraged. If necessary, identify subroutine issuing message.
- Identify significant parameters for major edit printout.
- Identify less significant parameters for minor edit or plot capability.

These requirements are from the section entitled 'BIT PATTERNS'.

- Bit map patterns should be used for high level logic or control functions to minimize the transfer of large amounts of binary data such as turning models off or on.
- Existing bit patterns should be used where appropriate and cannot be altered without INEL concurrence.

These requirements are from the section entitled ‘DATA BASE’.

- All data should be accessed directly from the common blocks. Accessing data by passing it on call statements is not acceptable.
- New models must have restart capability.
- Programmers should not resequence either the first or last variable of each common block saved to the restart file. This will seriously compromise the restart capability.
- Identify only the common blocks which are truly needed for restart capability. Unneeded common block storage can waste significant amounts of file space for each restart file.
- One of the acceptance tests for each new model, prior to it’s incorporation into the code, will be the ability to restart in the middle of a transient, with no change in results.
- New models must be able to utilize the time step repetition capability.
- Use predefined parameters as array dimensions and/or do loop indices, rather than ‘hardwiring’ values into the models.
- All common blocks should be called in ‘scdadv’. Each variable within a common block should be documented in a parallel “comment” common block.

8. REFERENCES

1. T. Heames et al., *VICTORIA: A Mechanistic Model of Radionuclide Behavior in the Reactor Coolant System Under Sever Accident Conditions,* NUREG/CR-5545, SAND90-0756, Rev. 1, December 1992.
2. C. M. Allison, C. S. Miller, and N. L. Wade (Eds.) *RELAP5/MOD3 Code Manual, Volumes I through IV,* NUREG/CR-5535, EGG-2596, DRAFT, June 1990.
3. C. M. Allison and G. H. Beers, "*Comparisons of the SCDAP Computer Code with Bundle Data Under Severe Accident Conditions*," Seventh International SMIRT Conference, Chicago, IL, August 22-26, 1983.
4. H. Jordan and M. R. Kuhlman, TRAP-MELT2 User's Manual, NUREG/CR-4205, BMI-2124, May
5. 1985.E. C. Lemmon, *COUPLE/FLUID A Two-Dimensional Finite Element Thermal Conduction and Advection Code,* EGG-ISD-SCD-80-1, February 1980.

Appendix A. CVI INPUT SUBROUTINE

A.1 PURPOSE

The CVI subroutine processes data in fields which are self delimiting and which can contain data in integer and floating point decimal, hex, octal, and three hollerith forms. The principal advantage of this input routine over the standard Fortran formatted input routine is that the input data need not be entered in predetermined column. This allows the design of input data forms to be simpler and the data entry easier. The subroutine also facilitates the handling of data input lines with variable numbers of words per input line. Both single and double precision entries are available.

A typical example of data converted by the subroutine might be two input lines entered as follows:

```
105,318,105+2,-14,-25-a*THIS IS A COMMENT
10 25    2.lE-3 -1.4+5 245.65, TEST
```

The subroutine would convert the input fields 105 and 318 to their binary integer equivalents, 105+2 would be converted to the binary floating equivalent of 10.5. In a similar fashion, -14 would be converted to a negative binary integer and -25-4 would be converted to the binary floating point equivalent of -0.000025. The asterisk indicates the end of the input fie1ds to be converted and the remainder of the input line can be used for comments. The second input line contains the integers 10 and 25, the f1oating point numbers 0.0021 -140,000, and 245.65 and the hollerith quantity TEST.

A.2 DATA FIELD DESCRIPTION

Input information to be converted is grouped into fields. The conversion process uses a left to right scan of the input line columns. The scan begins with a search for the beginning of a field, where the beginning of a field is the first non-blank character. Partial conversion occurs during the scan of the field and the conversion is completed after detecting the end of the field. If errors are found, a number is returned indicating a column within the field if detected during the scan of the field or the terminating column if detected during the completion of the conversion. When the conversion of a field is completed, thc search for a new field resumes. Thus, fields may be separated by any number of blanks. If an asterisk or a dollar sign is encountered during the search for a new field, the scan is terminated and the remainder of the input line may contain comments. If a field is terminated by an asterisk or a dollar sign, the conversion of the field is completed and the remainder of the field may be comments. The decimal, hex, octal, and Hollerith fields can be mixed in any order on a data line, however, the rules governing the conversion are listed separately for each type.

A.2.1 Decimal Fields

1 Each field generates one binary word. The field is initiated by the first non-blank character. A field is terminated by either a blank [note- one exception in (4) below], a comma, an asterisk, or a dollar sign.

2 The sign of the decimal number if present occupies the first column of the field. a plus sign is assumed if no sign is present.

3 An integer is indicated if none of the following is present in a field:

a a decimal point

b a sign, called an exponent sign, following a digit in a field and followed by an integer number

c an “E” or “D’ with or without a following exponent sign (+, -, or &) following a digit in a field or followed, by an integer number.

4 A floating point quantity is indicated if any of the conditions listed in (3) above is present. The digits proceeding the exponent sign, “E”, or “D” form the fractional part of the floating point number. If no decimal point is present, it is assumed to be in front of the first digit of the fraction. The exponent sign and following number are the power of ten by which the fraction is multiplied. The exponent field need not be present if a decimal point is used. One blank immediately following an “E” or “D” is treated as if it were a plus exponent sign. This is the only exception to the rule that a blank terminates a decimal field.

5 Note that the decimal field format is such that data input lines generated by Fortran routines using I. E, D, F, or G conversion can be read as long as at least one blank or a comma separates each field. Also this format is compatible with the input conventions of Fortran except:

a the restrictions of no blanks between characters in the number

b the convention of an assumed decimal point at the beginning of the fractional part of a number if no decimal point is present

c the placement of data on the input line with arbitrary spacing between fields.

6 The range of integer data must be such that it can be stored in a four-byte (single precision or 32 bit) word~ even if the~ subroutine was called through a double precision entry. Thus, the integer quantity must be equal to or greater than -2^{31} and less than $+2^{31}$. However, when called through a double precision entry, the double precision word (64 bits) appears as a correct double precision integer in that the high order 32 bits are set to zero if the integer is positive and are set to ones if the integer is negative. The magnitude of the fractional part of the floating point quantity with the decimal point removed must be less than 2^{64}. The magnitude of the floating point quantity must be within the floating point range of the 360, approximately 10^{-78} to 10^{+75}. Floating point values beyond these values are handled by the floating point overflow and underflow routines supplied by the system. A floating point quantity is converted as a double precision floating point number; when called through a single precision entry, only the high order 32 bits are stored. Both integer zero and floating point zero, regardless of sign, are stored as plus zero, that is as all zero bits. Both integer and floating point zeros will thus test zero in either an integer or floating point test. Hence, +0, -0, +0+0 all generate the same binary quantity. (Note that the machine is incapable of representing integer minus zero, but can represent a floating point minus zero.)

7 Leading zeros are ignored. However, zeros preceding non-zero digits are significant in a floating point number without a decimal point because of the assumed decimal point in front of the first zero.

8 Errors detected in decimal fields include illegal characters, multiple decimal points, a decimal point in the exponent field, sign placement errors, multiple “E” or “D”, no digits in the fraction or exponent fields, or magnitude errors described in (6) above, except for floating point overflow and underflow. A comma may be used only to terminate a field; a comma preceded by a blank is an error. Commas entered in adjacent columns may not be used to enter a zero.

A.2.2 Hexadecimal Fields

1 Each field generates one binary word. The hexadecimal (base 16) field is initiated by the first two non-blank characters being “/X”. The field is terminated by either a blank, a comma, an

asterisk, or a dollar sign.

2 The digits allowed in a hexadecimal field are the decimal digits O through 9, and the alphabetic characters A through F having values 1 through 15. No signs are allowed. The converted quantity is assumed to be a hexadecimal integer. Leading zeros are ignored.

3 Hexadecimal numbers can be considered a short, convenient form for binary information, and arc usual1y used to enter masks for packing and unpacking. Each hexadecimal number represents four binary bits. When entering numerical data, minus quantities must be entered in their two's complement form.

4 A field can contain eight hexadecimal digits (32 bits) When called through a single precision entry and 16 hexadecimal digits (64 bits) when called through a double precision entry.

5 Examples of hexadecimal fields are /XFFOO, /XFFFFFFF0, and /X123456789ABCDEF. The last number could only be used in a double Precision entry to the subroutine.

6 Illegal characters and magnitude violations are detected as errors.

A.2.3 Octal Fields

Octal (base 8) fields are similar to the hexadecimal fields with thc following differences.

1 The octal field is initiated by the first two non-blank characters being character "/O".

2 The allowed digits are 1 through 7.

3 The size of the binary quantity must not exceed 32. bits in single precision entries and 64 bits in double precision entries. Thus, the 11th octal digit (from the right) must be less than 4 in single precision entries and the 21st octal digit must be less than 2 in double precision entries.

Examples of octal fields are /O774, /O37777777770, and /Ol234567.

A.2.4 Hollerith Fields

Three types of Hollerith fields are allowed, two of which are identical to the Hollerith fields used in Fortran FORMAT and DATA statements.

1 Each Hollerith field generates one or more binary words. Thc two Fortran like fields arc initiated by either thc first nonblank character being a "'", or by a "H'" immediately following an integer decimal number. Fields started by a "'" arc terminated by a "'"; fields started by a "H" extend for the number of columns indicated by the decimal number preceding the "H". The number of columns indicated must not extend beyond the 80 columns of the input line. The end of an input line cannot terminate the Hollerith field started by an ""'

   In these types of Hollerith fields, an error is indicated if a blank or a comma does not follow the field unless the field is terminated in column 80.

2 The third type of Hollerith field is indicated by the first non-blank character being other than a digit, a plus or minus sign, a comma, an asterisk, a dollar sign, or a slash. Thus, the first character being an alphabetic character (or any other character except those just listed) indicates the third type of Hollerith field. Thc field is terminated by a comma or a blank.

3 Al1 254 ASCII characters are allowed in the first two types of Hollerith fields; the third type is restricted only in the first character of the field, and of course the terminating characters cannot

be entered. The restrictions on the first character exist because these characters specify other formats or terminate the scan. For example, a sign or digit indicate a decimal field. The field 100K is an erroneous decimal field, but ‘100K’ is a valid Hollerith field. An asterisk or a dollar sign in a Hollerith field does not terminate the scan of the input line. In fields initiated by a “’”, the character “’” can be entered in the Hollerith field and distinguished from the delimiting “’”, by putting two “’”s in the Hollerith field for every one desired. As examples of Hollerith fields and rules for “’”s, the character fields DON’T and ’DON’T’ are written using the first two Hollerith types as

5HDON’T,’DON’’T’ 7H’DON’T’ ’’’DON’’T’’’.

The Hollerith field DON’T can be written simply as DON’T when using the third type of Hollerith field, but the field ’DON’T’ cannot be entered in that Hollerith field type. A single “’” on column 80 or the “’” of an odd number of consecutive “’”s ending on column 80 is treated as a terminating “’”.

4 The first character is stored in the left most byte of the storage location. Succeeding characters are stored in consecutive bytes. Four characters per word are stored in single precision entries and eight characters per word are stored in double precision entries. If necessary, blanks are inserted to pad out the last single or double word.

A.3 SUBROUTINE USAGE

The entry names are CVIRC, CVIC, DCVIRC, and DCVIC. The standard subroutine linkage conventions are followed. The Fortran calling sequences are

```
CALL CVIRC (BCD,BIN,IC,NUM,NERR)
CALL CVIC(BCD,BIN,IC,NUM,NERR)
CALL DCVIRC(BCD,BIN,IC,NUM,NERR)
CALL DCVIC(BCD,BIN,IC,NUM,NERR).
```

In CVIRC and DCVIRC, an input line is read from Fortran unit 5 into the 80 byte array BCD. The input line is read using the standard Fortran input routine with 20A4 format. If the end of data is encountered, the program is terminated by the standard Fortran end of data set exit. In CVIC and DCVIC, the input line information is assumed to be already stored in BCD. BCD must be on a word boundary.

When called through CVIRC or CBIC the converted binary information is stored into consecutive locations in the single precision, array BIN starting at BIN(l). When called through DCVIRC or DCVIC, the converted binary information is stored into consecutive locations in the double precision array BIN starting at BIN(l).

For each converted binary quantity stored in BIN(1), the following code is stored into the corresponding half-word integer array IC(i).

Table A-1: Codes Returned from CVI

Code	Meaning
0	Binary quantity is an integer or floating point zero entered in a decimal field.
1	Binary quantity is a non-zero floating point quantity entered in a decimal field.
2	Binary quantity is a non-zero floating point quantity entered in a decimal field.
3	Binary quantity entered in hexadecimal field.

Table A-1: Codes Returned from CVI (Continued)

Code	Meaning
4	Binary quantity entered in octal field.
minus non-zero quantity	Binary information entered in Hollerith field.

Hollerith fields can generate more than one binary word. The code word for the first binary word of a Hollerith field contains the number of characters in the field as a negative quantity. Succeeding code words generated from thc Hollerith field have a code 4 greater than the preceding code in single precision or 8 greater than the preceding code in double precision. Thc code for the last binary word of a Hollerith field contains a number less than zero and equal to or greater than -4 single precision and -8 in double precision. The code numbers do not include that blanks that may have been added to fill out a word. Thus if an input line contains the fields 'ABCDEFGHIJ' and 'KLM', a single precision entry to CVI would give

```
BIN(l) = ABCD     IC(l)=-10
BIN(2) = EFGH     IC(2)=-6
BIN(3) = IJ       IC(3)=-2
BIN(4) = KLM      IC(4)=-3.
```

(Note that the binary quantity actually stored in BIN(l) would be ClC2C3C4 in hexadecimal notation where Cl is the bit code for the character "A", C2 for "B", etc.)

A double precision entry to CVI for the same information would, give

```
BIN(l) = AECDEFGH          IC(l)=-l0
BIN(2) = IJ                IC(2)=-2
BIN(3) = KLM               IC(3)=-3
```

The subroutine returns in NUM the number of binary quantities converted until either a normal scan termination or a detection of an error. NUM would contain zero if either a blank input line or an input line in which the first non-blank character is an asterisk or dollar sign is read.

NERR is used both as an input quantity and as an output quantity. If NERR is zero on input and an error is detected, a three line error statement is printed on Fortran unit 6 using the standard Fortran output routines. The first line is blank; the second line contains, ERROR IN CARD BELOW. AT COL. xx; the third line lists the input line in error. If NERR is non-zero on entry, no error statement is printed on detection of an error. On return from CVI, if no error is detected, NERR contains zero. If an error is detected, NERR contains the column containing the error.

Note that the BIN array can have a mixture of real and integer information; thus, the Fortran variables used for this information must be chosen such that no undesired integer-to-real or real-to-integer conversion occurs. Equivalence statements can be used to avoid this problem. Also a common user error when inputting data with this routine is to enter a floating point number in integer form or visa versa. This can be detected or corrected by testing the contents of the IC array.

When used in single precision, typical declaration statements used with CVI are

```
DIMENSION BCD*4(20), BIN*8(40), IC*2(40), IBIN*4(40)
EQUIVALENCE (BIN(l),IBIN(l)).
```

The ith variable can be obtained from BIN(i) if real and from IBIN(i) if integer.

When used in double precision, typical declaration statements are

```
DIMENSION BCD*4(20), BIN*8(40), IC*2(40), IBIN*4(2,40)
EQUIVALENCE (BIN(l),IBIN(l,l)).
```

The ith variable can be obtained from BIN(i) if real and from IBIN(2,i) if integer.

The subroutine package includes the driver routine and data input lines used to test the subroutine and to illustrate its use.

A.4 RESTRICTIONS AND CAPABILITIES

The subroutine should detect all input line format and magnitude errors except floating point overflow and underflow. Machine interrupts can occur in the case of floating point overflow and underflow, and addressing and specification errors due to incorrect subroutine call parameters.

Appendix B. COMMON BLOCK DESCRIPTIONS

B.1 Comdeck CMPDACC

dialn	surge line diameter.
lnelv	surge line elevation drop.
thick	tank wall thickness.
lnlen	surge line length.
ttank	tank wall temperature.
ttanko	tank wall temperature, old time.
thcnd	noncondensible thermal conductivity.
htcap	tank wall specific heat capacity.
rhot	tank wall density.
htxr	heat transfer flag.
qtank	heat transport to noncondensible from all sources.
vdm	dome volume (noncondensible).
vdmo	dome volume (noncondensible), old time.
vliq	liquid volume (tank and surge line).
vliqo	liquid volume (tank and surge line), old time.
rhon	noncondensible density.
rhono	noncondensible density, old time.
tvapo	noncondensible temperature, old time (tempg is new time).
cvnit	noncondensible specific heat capacity, cv.
vtank	tank volume.
betav	steam saturation coefficient of expansion.
dpd	variable used in solution matrix (right hand side).
dpddp	variable used in solution matrix (left hand side).
ahfgtf	heat of vaporization at the liq temp.
ahfgtg	heat of vaporization at the vap temp.
avgtg	vap specific vol at the vap temp.
avfgtf	vg - vf at the liq temp.
ahftg	liq enthalpy at the vap temp.
acpgtg	vapor specific heat, cp, at the vap temp.
acvgtg	vapor specific heat, cv, at the vap temp.
aviscn	noncondensible viscosity.
acpnit	noncondensible specific heat, cp, at the vap temp.
atank	tank cross sectional area.
diamtk	tank diameter.
ahgtf	vap enthalpy at the liq temp.
dztank	tank elevation change.
qtanko	heat transport to noncondensible from all sources (old time).
dmgdt	time rate of change in dome vapor mass (equilibrium model).
claptf	clapyron coefficient at the liq temp.
acctrp(1)	accumulator trip number.
acctrp(2)	trip offset during input, index during transient.
gasln	length of gas above the surge line.
gaslno	length of gas above the surge line, old time.
tklen	length of the tank above the surge line.

B.2 Comdeck CMPDATC

ncmps	number of components.
cmpnum	component number.
cmpnam	component name (alphanumeric).
cmptyp	component type.
cmplen	length of component data.
nvc	number of volumes in component.
nvcn	position number of first volume of this component.
nvco	offset to first volume of this component in volume block.
njc	number of junctions in component.
njcn	position number of first junction of this component.
njco	offset to first junction of this component in junction block.
cmpopt	for all components, velocity/mass flow switch (1 bit).
	for turbine component, drain flag (4 bit), stage type (16-8 bits).
	for valve component, ? (4 bit), ? (8 bit), open/close flag (16 bit).
	for accumulator component, new time accumulator active flag (4 bit), new time accumultor empty flag (8 bit), new time volume active flag (16 bit), old time values of the previous three bits (32, 64, and 128 bits).
	for pump component, pump stop flag (4 bit), pump stopped flag (8 bit), shaft connected flag (16 bit), negative pump velocity flag (32 bit), single phase referral flag (64 bit), single phase referral satisfied flag (128 bit), head and torque multiplier referral flag (256 bit), head and torque multiplier referral satisfied flag (512 bit), two phase referral flag (1024 bit), two phase referral satisfied flag (2048 bit), motor torque referral flag (4096 bit), motor torque referral satisfied flag (8192 bit), pump speed table referral flag (16384 bit), pump speed referral satisfied flag (32768 bit), octant number (shifted left 24 bits).
ncttrp	trip number.
ncttrx	offset in trip block, then index to trip duringtransient routines.
nctalf	alphanumeric part of variable request code.
nctdpv	numeric part of variable request code. nctpc(1) 0 if common block, block number if dynamic block nctpc(2) offset, then index in transient routines.
nctblt	table type.
nctble	number of items/entry.
nctbtn	total number of items.
nctbtx	last index.
cmptbl	table entries.

B.3 comdeck cmpdtvc

definitions to all valves:

vlvnm	valve type.
atlast	normalized full open valve area.

definitions to check valves:

pcv	back pressure to close valve.
aleak	normalized area of valve leakage.

definitions to trip valves:

vlvtrp	trip number.

definitions to inertial valve:

theta	valve disk angular position.
thetao	valve disk angular position, old time.
mintht	minimum valve disk angular position.
maxtht	maximum valve disk angular position.
moment	mass moment of inertia of valve disk.
omega	valve disk angular velocity.
omegao	valve disk angular velocity, old time.
lngvlv	length of valve disk moment arm.
rdsvlv	valve disk radius.

definitions to motor valve:

opntrp	open trip number.
clstrp	close trip number.
vlvslp	rate of change of normalized stem position.
vlstm	normalized stem position.
vlstmo	normalized stem position, old time.
vlvtbl	table number of “normarea” table.
vlvcon	multiplier for converting csubv to form loss.
ncvtbl	variable used for table interpolation as follows: unused (15 bits), number of items/entry (15 bits), total number of items (15 bits), current subscript (15 bits).
cvtbl	table enteries (sets of three starting with normalized stem position, fjunf, fjunr).

definitions to servo valve:

opntrp	control variable number.
clstrp	used for control variable information.
vlstm	normalized stem position, controlled by control variable.
vlvtbl	table number of “normarea” table.

B.4 comdeck comctlc

ncoms	number of common control slots.
nfiles	number of dynamic file slots.
comdat	index relative to fa of first word of common block to be saved.
comdln	length of common block to be saved.
filid	ftb filid for dynamic storage files.
filsiz	length of dynamic file.
safe1	not written on restart file, provided to allow length checking when reading from restart file. Used for timer argument when timing transient subroutine execution times.
newrst	true if a restart problem, used during input processing.
filndx	index of dynamic file in fast or ftblcm block.
filflg	flag for dynamic file on disk to be written at complete restart (bit 1), flag to write common block file at complete restart (bit 4).
filid(1)	input data and work scratch space during advancement.
filid(2)	time step control block.
filid(3)	component description block.
filid(4)	hydrodynamic volumes block.
filid(5)	hydrodynamic junctions block.
filid(6)	thermodynamic property file.

filid(7)	interactive input and output variable storage.
filid(8)	heat structure geometry and temperature block.
filid(9)	heat structure material property storage.
filid(10)	table of inlet and outlet junctions for each volume.
filid(11)	general table storage.
filid(12)	minor edit file.
filid(13)	time dependent volumes and junctions pointers.
filid(14)	table of heat structures and data for each volume.
filid(15)	plot heading and control information.
filid(16)	minor edit control, save area, and labels.
filid(17)	plot record buffer.
filid(18)	trip block.
filid(19)	internal plot file control information.
filid(20)	file for statistics during advancement.
filid(21)	reactor kinetics data.
filid(22)	2d plot requests and specifications.
filid(23)	plot comparison data tables.
filid(24)	steady state block for statistics counters and uo.
filid(25)	volume data needed for a moving system.
filid(26)	plot data for the internal plot routines.
filid(27)	control system block.
filid(28)	component indices in normal (input) order.
filid(29)	fusion kinetics data.
filid(30)	hydrodynamic system control information.
filid(32)	reflood rezoning model storage space.
filid(33)	user supplied plot record variable requests.
filid(34)	fission product data.
filid(35)	fixed list vectors.
filid(36)	SCDAP data.
filid(37)	steady state initialization check file.
filid(38)	ile for radiation heat transfer.
$if	def,expnum,1.
filid(39)	file for numerics experimentation.
filid(40)	file for sparse matrix strategy.

B.5 Comdeck couplc

common block for COUPLE debris bed model. saved on restart file.

afrdeb	atomic fractions of debis bed.
dtcoup	COUPLE time step. used with ntsc to determinewhen to call COUPLE.
hgtdeb	height of debris bed.
pdmave	sum of particle diameter * mass of material slumped.
pmufot	multiplier for COUPLE mesh fission power or total power.
pmufdk	multiplier for COUPLE mesh fission product decay power.
pmuadk	multiplier for COUPLE mesh actinide decay power.
qfrspi	fraction of fission product decay heat for each species.
porave	sum of porosity * mass of material slumped.
powave	sum of dt * power of material slumped.
powdmx	maximum power density in debris bed.
pdbtot	total power density in the COUPLE mesh.

stmass	steam production rate of debris in COUPLE mesh, kg/s.
tave	sum of temperature * mass of material slumped.
timc	selap time for next call to COUPLE.
tmpdmx	maximum temperature in debris bed.
twalmx	maximum temperature of structure in COUPLE mesh.
tmpdav	average temp. of debris bed (material 1).
twalav	average temp. of structure (materials 2, 3, and 4).
wtmdeb	total mass in debris bed.
ws	mass of each material which has slumped into the COUPLE mesh since last COUPLE call. the order =zr, u, ss, ag, b4c, uo2, zro2, al, li, cd.
wst	total mass of each material which has slumped into the COUPLE mesh.
icoup	COUPLE status indicator - 0 = COUPLE model not used. 1 = COUPLE model used, no debris bed yet. 2 = COUPLE model used, debris bed is forming.
invc	index offset of data for volume ncvolc.
itsc	number of selap hydraulic time steps since last call to COUPLE.
mcostr	pointer to start of COUPLE data in array a for mesh mcp.
ncdtsl	no. of selap time steps since last significant slump.
ncpowi	indicator for type of power source -0 = constant power from slumping material at time of slumping, 1 = power is determined by RELAP5 reactor kinetics power in the associated SCDAP components, 2 = power is determined by power other than RELAP5 reactor kinetics power in the associated SCDAP components.
ncslp	indicator for whether or not any material has slumped into COUPLE mesh this COUPLE time step. 0 = no, 1 = yes.
ncvolc	number of the RELAP5 volume to receive COUPLE debris.
npowdm	node no. of current maximum debris bed power density.
ntmpdm	node no. of current maximum debris bed temp.
ntsc	maximum number of selap hydraulic time steps to use per COUPLE time step.
ntwalm	element no. of current maximum structure temp.
nmcpwr	number of COUPLE mesh to be written to output file.
kpr	interval of RELAP5 time steps to write output file.
nrecno	next record number in COUPLE output file.

B.6 Comdeck eccmxcc

cmpphi	injection angle of ECC.

B.7 Comdeck fastc

arrays define dynamic pool area. fa and ia are equivalent floating and integer areas.

B.8 Comdeck fpfastc

Fission Product Fixed data block

fpinpo	offset to fission product input data.
fpactv	fission product transport active flag.
fpdbgr	fission product debug flag.

fpnsys number of fission product systems.
fpsyso offset to first fission product system.
fpnbin number of aerosol bins modeled.
fpnsp number of species tracked.
fpsysk fission product system data skip factor.
fpvolk fission product volume data skip factor.
fpsrfk fission product surface data skip factor.
fpspvk fission product species skip factor within volume data.
fpspsk fission product species skip factor within surface data.
fprtol relative convergence criteria.
fpatol absolute convergence criteria.
fpsp species names which are tracked.
***
Fission Product System data block.
fpsysn hydrodynamic system number (ordinal) of fp system.
fpnvol number of volumes in fission product system.
fpvolo offset to first volume in fission product system.
fpsysm species mass in system.
***
Fission Product Volume data block.
fpvoln hydrodynamic volume number of fission product volume.
fpnsrf number of surfaces in fission product volume.
fpsrfo offset to first surface in fission product volume.
fpstep last successful time step.
fpsrc species source rate in volume.
fpvtm species total mass in volume.
fpstm species total mass on all surfaces in volume.
fpliq species liquid mass in volume.
fpvap species vapor mass in volume.
fpbin species aerosol bin mass (each bin) in volume.
***
Fission Product Surface data block.
fpsrfn hydrodynamic surface number of fission product surface.
fpsflr left or right side volume indicator.
fpsfmt surface material type.
fpsfto offset to surface temperature data.
fpsfar surface area.
fpsfan surface angle.
fpsfra relative surface area.
fpsfbs decay heat due to beta on surface.
fpsfgs decay heat due to gamma on surface.
fpsfbv decay heat due to beta in volume (as a fraction of the gamma decay heat).
fpsfmc species condensate on surface.
fpsfma species absorbate on surface.
fpsfmp species precipitate on surface.

B.9 Comdeck genrlc

ctitle contains title card of case, time, and date.

ptitle	contains program version identification and title.
uniti	units for input, si if true, british if false.
unito	units for output, si if true, british if false.
safe3	same purpose as safe1.

B.10 Comdeck htrcomc

In the comments below, input means that the value is set before calling htrc1, computed means that it is set inside htrc1 and is for communication to lower level routines, and output means that the quantity is returned by htrc1.

axpf	axial power peaking factor (used for chf) (input).
beta	coefficient of thermal expasnsion of appropriate phase (computed).
chf	critical heat flux (output).
chfmul	critical heat flux table lookup multiplier (output).
cps	specific heat of appropriate phase (computed).
dtsat	wall temperature minus saturation temperature (computed).
fstrat	percent of liquid stratification (computed).
g	total cell centered mass flux (input).
gabs	absolute value of total centered mass flux (input).
gamw	direct wall flashing or condensation (output).
ggasa	cell centered vapor mass flux (input).
gliqa	cell centered liquid mass flux (input).
gridz	distance to grid spacer (used for chf) (input).
gridk	grid spacer pressure loss coefficient (used for chf) (input).
hfg	heat of vaporization (enthalpy difference between saturated vapor and saturated liquid) (input).
htcf	heat transfer coefficient to liquid (output).
htcg	heat transfer coefficient to vapor (output).
htcoef	total heat transfer coeffieient (output).
htdiam	heated equivalent diameter (input).
htlen	heat transfer length from some position (used for chf) (input).
htsa	heat transfer surface area (input).
qffo	heat flux to liquid (output).
qfgo	heat flux to vapor (output).
qfgox	left (1) and right (2) heat flux to liquid (output).
qfluxo	total heat flux (output).
tf	volume liquid temperature (computed).
thcons	thermal conductivity of appropriate phase (computed).
tsat	saturation temperature corresponding to partial pressure of fluid (input).
tw	wall surface temperature (input).
viscs	viscosity of appropriate phase (computed).
htopta	heat transfer package option number.
ibundl	rod bundle flag (computed).
irwt	vertical/horizontal flag (computed).
iv	volume index (input).
mode	mode of heat transfer (output).

B.11 Comdeck htrflbc

nrflht	number of heat structure-geometries designated for reflood-rezoning calculations.
lhtrfl	offset to ihtptr for first heat structure in a heat structure designated for reflood.
inxrfl(1)	offset/index to heat structure dependent data.
inxrfl(2)	offset/index to axial mesh point dependent data.
inxrfl(3)	offset/index to temperature data.
inxrfl(4)	offset/index to mesh point positioning data.
iglrfl(1)	maximum number of axial levels.
iglrfl(2)	current number of axial levels.
qfchfn	critical (maximum) heat flux.
tchfqf	temperature corresponding to qfchfn.
trewet	quench or rewet temperature.
qfhtcn	critical heat transfer coefficient.
strgeo	heat structure-geometry which reflood set belongs to (on restart this word is set to 0 or negative if the heat str is deleted or overlayed in rhtcmp).
tmprfo	old time temperature array.
tmprfn	new time temperature array.
htlenz	axial mesh points coordinate.

B.12 Comdeck htscrc

hte	temporary storage for left part of tridiagonal matrix.
htf	temporary storage for right hand side of heat conduct. eqs.
httc	temporary storage for thermal conductivity.
htvhc	temporary storage for volumetric heat capacity.

B.13 Comdeck htsrcmc

htfixa	number of mesh size independent words for each heat structure.
htgskp	skip factor for geometry data.
nhtstr	number of heat structures.
ihtptr	offset to heat structure.
	The following are for each structure.
htstno	heat structure number.
htopt	steady state flag during input and reflood rezoning indicator during transient (1 bit); input error flag (2 bit); initialization flag (4 bit); geometry referral flag (8 bit); geometry satisfied flag (16 bit); temperature referral flag (32 bit); temperature referral satisfied flag (64 bit); left extra boundary condition card entered (128 bit); right extra boundary condition card entered (256 bit); reflood side, 0 = left, 1 = right (512 bit); geometry type shifted left 24 bits.
htxft	offset to end of time step temperatures.
htxit	offset to beginning of time step temperatures.
htgmr	heat structuregeometry referral number.
htgom	offset to heat structure geometry data.
htsrt(1)	heat source type.
htsrt(2)	heat source offset for general table or control system power source; heat structure-geometry referral number for initial temperatures. for htbvc, htbvo, htbnt, htbntr, htbnts, htstyp, and htmod, quantity(1) is for left side, quantity(2) is for right side.

htbvc	boundary volume input code.
htbvo	boundary volume offset during input, index during transient or general table offset.
htbnt	boundary type input.
htbntr	reduced boundary type.
htbnts	subroutine number (bit pos. 8-11); option number (bit pos. (1-7); or table ordinal from boundary type.
htstyp	heat transfer surface type during input.
htmod	heat transfer mode number during transient. This quantity and the preceeding quantity occupy the same location.
htgap(1)	gap deformation model reference volume number.
htgap(2)	gap deformation model reference volume offset.
htnmpt	number of mesh points.
htrflg(1)	reflood flag.
htrflg(2)	offset for trip when trip number in htrflg(1).
htnaxl	axial rezoning limit number.
htnusr	number of heat structures with this geometry.
htfctr	source multiplier.
htsrfo	area at left boundary.
htsrfn	area at right boundary.
htrnro	heat transfer rate at left boundary (new).
htrnrn	heat transfer rate at right boundary (new).
htrnso	heat transfer rate at left boundary (old).
htrnsn	heat transfer rate at right boundary (old).
htcffo	heat transfer coefficient to liquid at left boundary.
htcffn	heat transfer coefficient to liquid at right boundary.
htftro	left direct source factor.
htftrn	right direct source factor.
htpowo	old power value.
htpown	new power value.
httots	integral of the source distribution over space.
htimeo	time at beginning of advancement.
htdt	time increment.
hthdmo	left side heated equivalent diameter.
hthdmn	right side heated equivalent diameter.
htchfo	left side critical heat flux.
htchfn	right side critical heat flux.
htvatp	volume averaged temperature.
htdtmo	left side old (twall - twater).
htdtmn	right side old (twall - twater).
htbcao	left boundary condition value, usually heat transfer coefficient.
htbcan	right boundary condition value, usually heat transfer coefficient.
htbcco	left boundary condition value, usually sink temperature related quantity or heat flux.
htbccn	right boundary condition value, usually sink temperature.
related	quantity or heat flux.
gprouf	fuel surface roughness + cladding surface roughness.
gpudis	cladding creepdown radial displacement - radial displacement due to fission gas induced fuel swelling and densification.
htrado	radius at left boundary.
htradn	radius at right boundary.
htiscr	index for scratch space during transient advancement.

The following are for each mesh point or mesh interval.

httmp	temperature in heat structure.
areao	surface weight at left boundary.
arean	surface weight at right boundary.
htsrc	pace dependent source factor times volume weights at each mesh point.
htcmp(1)	composition number.
htcmp(2)	composition offset.
htavwt	volume weights for average volume temperature calculation at each mesh point.
htsrwt	surface weights at each mesh interval.
htrvwt	right volume weights at each mesh interval.
htlvwt	left volume weights at each mesh interval.
gpintp	gap initial pressure at input, initial presuure /temperature of reference volume (top of the core) after initialization.
gprinc	radial interval width up to the gap plus inner and outer radii of cladding.
chfmuo	critical heat flux multiplier, left-hand side.
chfmun	critical heat flux multiplier, right-hand side.
htlnfo	forward direction heated length from inlet, left-hand side.
htlnfn	forward direction heated length from inlet, right-hand side.
htlnro	reverse direction heated length from inlet, left-hand side.
htlnrn	reverse direction heated length from inlet, right-hand side.
grdzfo	forward direction length from grid spacer, left-hand side.
grdzfn	forward direction length from grid spacer, right-hand side.
grdzro	reverse direction length from grid spacer, left-hand side.
grdzrn	reverse direction length from grid spacer, right-hand side.
grdkfo	grid spacer forward form loss coefficient, left-hand side.
grdkfn	grid spacer forward form loss coefficient, right-hand side.
grdkro	grid spacer reverse form loss coefficient, left-hand side.
grdkrn	grid spacer reverse form loss coefficient, right-hand side.
apfaco	axial power peaking factor, left-hand side of heat structure (local flux/average flux from start of boiling to local point).
apfacn	axial power peaking factor, right-hand side of heat structure.

The following are for metalwater reaction.

oxti	new oxide thickness on inside of cladding (m).
oxto	new oxide thickness on outside of cladding (m).
h2gen	new hydrogen generated (kg).
oxtio	old value of inside oxide thickness (m).
oxtoo	old value of outside oxide thickness (m), (can be input as w2 on the 1cccg003 card in rhtcmp).
h2geno	old value of hydrogen generated (kg).
imw	packed word, clad ruptured flag, metal-water correlation used, and node number of inside clad mesh pt. right 9 bits = radial mesh pt number of inside of clad. 512 bit = on if rupture has occurred. 1024 bit = on if loss coefficients have been changed. 2048 bit = on if plastic strain to be calculated (input). 4096 bit = on if loss coef to be altered at rupture (input). 8192 bit = on if mwr being calculated on inner surface. 16384 bit = on if mwr being calculated (set by input).

The following are for clad balloning and rupture.

gapwd	gap width for the gap conductance and ballooning case.
cladex	clad expanded outer radius (negative if it has burst).
hetrat	is the cladding heatup rate.
strnpl	is the maximum plastic strain.
stanto,stantn	left, right side Stanton numbers.
peclo,pecln	left, right side Peclet numbers.

B.14 Comdeck intracc

intrno	number of interactive variables.
intrla	variable name of input quantity or label of output quantity.
intrcv	conversion factor to si units.
intrva	current value of input quantity.
intrni	input variable card number.

B.15 Comdeck invtblc

invofs	offset used to compute indexes stored in invvnx.

the following occur for each volume.

invcnt	number of junctions connected to each volume.

the following three words follow for each junction connected to a volume.

invjun	reverse coordinate direction flag, 0 if normal, 1 if reverse (1 bit); connection flag, 0 if inlet, 1 if outlet (2 bit); from/to flag, 0 if from, 1 if to (4 bit); cross flow flag, 0 if normal, 1 is special half cell volume (8 bit).
invvno	position number of junction in junction block.
invvnx	index of junction.

B.16 Comdeck iparmc

***** the following are pointers to the indicated arrays within the large COUPLE a array**

i5	properties for the nummat materials defined (8*nummat).
i6	radii of nodes (numnp).
i7	z coordiates of nodes (numnp).
i8	i,j,k,l, and mtl for nodes (5*numel).
i9	input angles beta; always 0.0 (numel).
i10	flags for material in region (numel).
i20	function controls (3*nfnt).
i21	function tables (2*no. of tables).
i22	initial temps. (numnp*amult).
i27	power densities (numnp).
i28	vol. heat generation rates (numnp).
i29	tables for vol. heat gen. (numnp).
i40	convection heat transfer node numbers (ncev).

i41	convection heat transfer coeffs. or mults. (ncev).
i42	convection heat transfer coeff. tables (ncev).
i43	convection heat transfer temps. or mults. (ncev).
i44	convection heat transfer temp. tables (ncev).
i45	radiation heat transfer node numbers (nrad).
i46	radiation heat transfer coeffs. or mults. (nrad).
i47	radiation heat transfer coeff. tables (nrad).
i48	radiation heat transfer temps. or mults. (nrad).
i49	radiation heat transfer temp. tables (nrad).
i50	radiation heat transfer surface areas (nrad).
i51	convection heat transfer surface areas (ncev).
i60	tempset node numbers (nttem).
i61	constrained temps. or mults. (nttem).
i62	constrained temp. tables (nttem).
i70	porosities (numel).
i72	particle diameters (numel).
i74	zircaloy atomic fractions (numel).
i76	uranium atomic fractions (numel).
i78	stainless steel atomic fractions (numel).
i80	silver atomic fractions (numel).
i82	boron carbide atomic fractions (numel).
i84	uranium dioxide steel atomic fractions (numel).
i86	zirconium dioxide atomic fractions (numel).
i88	indicator of material type in elem n (numel).
0.0	no material.
0.05	mixture of material.
0.1	mostly aluminum.
0.2	mostly B4C.
0.3	mostly silver.
0.4	mostly stainless steel.
0.5	mostly zr.
>0.5	mostly UO2 and ZrO2.
i90	power density of debris in elem n (numel).
i99	node point volumes (numnp).
i100	element volumes (numel).
i102	material volumes without pores (numel).
i103	material volumes in elements (numel).
i104	indicators for inital debris (0) (imtl).
i105	indicators for debris in elements (numel) (0 = none, 1= partly full, 2= full).
i107	power (4*numel).
i108	debris volumes (4*numel).
i109	densities * specific heats (4*numel).
i110	energies (4*numel).
i111	indicators for elements full of debris (numel).
i112	temperatures for subr. etemp (not used) (numnp).
i113	array for subr. etemp (not used) (4*numel).
i114	array for subr. etemp (not used) (numnp).
i115	array for subr. etemp (not used) (4*numel).
icompt	components to which COUPLE nodes radiate (nrad).

idetop	top elements through which debris bed passes (maxi-1).
igmcpt	volumetric heat generations (ncev).
igmopt	past time step volumetric vapor generations (ncev).
ihtcto	total heat transfer coefficients (ncev).
inetco	offset to RELAP5 general table of power density (numnp).
inodpt	numbers of COUPLE mesh node that corresponds with i-th node in convection set (ncev).
inussn	rayleigh number of molten debris element (numel).
ipafp	amount of gas (aerosol) released from COUPLE element in time step (kg) (numel*max(1,nspdeb)).
ipbfp	mass of ith aerosol in kth bubble size at start/end of time step (kg) for each element (allow for 15 bubble sizes) (numel*max(1,nspdeb)*15).
ipxfp	mass of ith species vapor or non-condensible gas in kth bubble size released from the volume element during the time increment (max(1,nspdeb)*15).
ipfrto	multiplier for RELAP5 general table power density (numnp).
iptalf	aluminum atomic fractions (numel).
iptars	original values of areac array (ncev).
iptcdf	cadmium atomic fractions (numel).
iptsif	soil atomic fractions (numel).
ipteij	element number corresponding with i,j coordinates for lower left corner (maxj*maxi).
iptfml	fraction of element melted (numel).
iptfp	total mass of ith aerosol at start/end of time step for each element (kg) (numel*max(1,nspdeb)).
iptfpv	offsets for identifying RELAP5 volume numbers adjacent to convective nodes of debris bed for condensible aerosols (add to fpsrc term) (ncevr5).
iptgav	offsets for identifying RELAP5 volume numbers adjacent to convective nodes of debris bed for noncondensible aerosols (add to gaman term) (ncevr5).
ipthfs	rate of release of heat of fusion (numel).
ipthtm	input specified h.t.c. for convective nodes that model gap resistance (ncev).
iptiel	i coordinate of element n (numel).
iptihs	original values of ih array (ncev).
iptjel	j coordinate of element n (numel).
iptkfr	effective thermal conductivity, radial direction (numel).
iptkfz	effective thermal conductivity, axial direction (numel).
iptlif	lithium atomic fractions (numel).
iptmdt	rate of melting, fraction/sec (numel).
iptmet	indicator whether convection to water or internal liquefied debris (ncev).
iptmlt	melting temperature of element (numel).
iptspd	inventory (kg) of each fission product species for each element (numel*max(1,nspdeb)).
iptspl	fission product species to be tracked (max(1,nspdeb)).
ipttsm	sink temperature for convective corium heat transfer (ncev).
iqcopt	total heat inputs to R5 volumes from i-th COUPLE nodes in convection set (ncev).
iqfopt	past time step heat fluxes input to vapor (ncev).
iqolpt	past time step convection heat fluxes (ncev).
iqotpt	current convective heat fluxes (ncev).
iqwcpt	total heat inputs to vapor phase (ncev).

ir5pt	RELAP5 vols. connected to COUPLE convective nodes (ncev).
itsink	sink temperatures (ncev).
ivcnpt	RELAP5 volume offset indices for i-th convection nodes (ncev).
ivfspt	RELAP5 offset to fission product arrays (ncev).
ivrapt	RELAP5 offset for vols. connected to i-th nodes in radiation sets (nrad).
ivrdpt	RELAP5 volumes interfacing with COUPLE nodes that radiate (nrad).
m1	array used by fluid sol. (nftot).
m2	array used by fluid sol. (nftot).
m3	array used by fluid sol. (nftot).
m4	array used by fluid sol. (nftot).
m5	array used by fluid sol. (nftot).
m6	array used by fluid sol. (nftot).
m7	a(m2)*a(m6) (numnp).
m8	array used by fluid sol. (numel).
m9	array used by fluid sol. (numel).
m10	array used by fluid sol. (2*numel).
m12	array used by fluid sol. (4*numel).
m13	array used by fluid sol. (numnp).
m14	array used by fluid sol. (numnp).
m15	array used by fluid sol. (numnp).
n1	minimum j node numbers (maxi).
n2	maximum j node numbers (maxi).
n3	minimum i node numbers (maxj).
n4	maximum i node numbers (maxj).
n5	r values (nmax).
n6	z values (nmax).
n7	indicators for valid node at given i,j's (1= yes) (nmax).
n20	indices of materials with no internal gen. (ngenm+1).
n100	temperatures at last time step (numnp).
n101	temperatures at current time step (numnp).
n102	power (numnp).
n103	power generating volumes assoc. with node points (numnp).
n132	constrained temperatures (nttem).
n150	constrained temperatures (nttem).
iptnbu	pointer to number of bubbles in each size range in each element.
iratpo	ratio of power density to average power density (this is currently set to 1.0) (numel).
iwalce	mass of aluminum in elements (numel).
iwuoce	mass of uranium dioxide in elements (numel).
iwurce	mass of uranium in elements (numel).
ipwxl2	pointer, oxidized aluminum (kg/m2).
ipexlm	pointer, heat generation due to aluminum oxidation (w).
iph2lm	pointer, rate of hydrogen production due to aluminum oxidation (kg/s).
ipoxth	pointer, aluminum oxidation thickness (m).
ilayer	matrix for userdefined layers of elements for ATR configuration (25*25).
inelms	vector to keep track of the number of elements in each layer in ilayer above (25).

*** end of COUPLE a array pointers.

idf	default values flag for skipping COUPLE input (1 = yes).
idhint	flag to turn off convective heat transfer at nodes in fluid field until material relocation (1 = yes).

iflagc	flag for use of convection heat trans. model (1 = yes).
iflagr	flag for use of radiation heat trans. model (1 = yes).
iflagt	flag for use of input tempsets.
ifluid	flag for use of fluid solution model.
iov	fluid in/out boundaries.
istep	counter for number of times COUPLE has been called.
itran	indiator if ave. element temp. is ok (1 = yes).
itx	COUPLE input error flag (1 = error).
ivsol	fluid velocity solution flag.
iupwnd	fluid upwind flag.
last	last used location in a array, + 1.
lhead	spherical lower head modeling flag (1 = yes; 0 = no).
line	line counter for printout of COUPLE input.
matflu	material index of the coolant.
maxe	mesh plotting regions parameter.
maxi	maximum number of nodes in horizontal direction.
maxj	maximum number of nodes in vertical direction.
mband	COUPLE bandwidth.
mcond	space needed for COUPLE conduction solution.
mld	number of lower codiagonals in conduction equations.
mnpo	always 0.
mpp	mesh plotting parameter.
mud	number of upper codiagonals in conduction equations.
ncev	number of nodes in convective set * amult.
ncevr5	number of nodes in convective set.
neltop	number of elements through which top of debris passes.
nfnt	total number of function tables.
nftot	numvt * amult.
ngenm	number of materials with no internal generation.
niter	number of iterations used for COUPLE conduction sol..
nline	input indicator for reading 2 cards (-1 = yes).
nmax	maxi * maxj.
nmtl	number of material blocks.
nnli	maximum number of iterations in COUPLE solution.
npdtbi	number of RELAP5 power density general tables- 0 = none, 1 = 1, 2 = more than 1.
npdtbo	offset to RELAP5 power density general table if only 1.
npp	geometric code (0 = r,z; 1= x,y).
nrad	number of nodes in radiation set * amult.
nslph	nslphs(mcp) from common / slumpv /.
nspdeb	input number of fission product species to be tracked in debris bed.
nttem	number of tempsets * amult.
numel	number of elements in COUPLE mesh.
numhc	number of nodes in convective set.
nummat	number of different materials to be defined.
numnp	number of node points in COUPLE mesh.
numrc	number of nodes in radiation sets.
numtc	number of nodes in tempsets.
numvt	fluid in/out nodes.
nunqsw	indicator for doing unquench calculation (1 = yes).

ncpow	indicator for type of power source- (same as ncpowi in /coupl/).
natrop	indicator for ATR configuration option.
ncrlay	number of the current layer when using ATR config.
ncld	switch to tune on/off natural convection in liquified debris and modified conductance.
ncld	0, use modified K and natural convection.
ncld	1, turn off modified K (subroutine kpool).
ncld	2, turn off modified K and nat convection.
model	(subroutine dhcoef).

*endif

B.17 Comdeck jundatc

ijskp	junction skip factor.
njuns	number of junctions.
ij1	from volume input code.
ij2	to volume input code.
jc	choking flag (1 bit); time dependent junction flag (2 bit); reversed from volume connection flag (4 bit); reversed to volume connection flag (8 bit); no choking flag (16 bit); old time choking flag (32 bit); choking test flag for accumulator junction (64 bit); input flag (128 bit); abrupt area change flag (256 bit); two velocity-one velocity flag (512 bit); separator flag (1024 bit); stratified flow flag (2048 bit); from cross flow option (4096 bit); to cross flow option (8192 bit); cross flow flag (16384 bit); accumulator active flag (32768 bit); stratification flag (65536 bit); stratification input data (bit pos. 18-19); jet mixer flags (bit pos. 20-22); separator flags (bit pos. 23-25); unused (bit pos. 26); horiz-vert junction flag (bit pos.27); up-down junction flag (bit pos. 28); valve flag (bit pos. 29); second turbine junction flag (bit pos. 30).
ij1vn	from volume ordinal number.
ij2vn	to volume ordinal number.
junftl(1)	from pointer in output form without sign.
junftl(2)	to pointer in output form without sign.
ajun	area of junction.
athrot	ratio of orifice area to junction area.
arat(1)	mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. mixture is obtained by using sum of absolute value of phasic volumetric flow rates. 1 is for “from” volume.
arat(2)	same as arat(1), except 2 is for “to” volume.
diamj	diameter of junction.

****** warning: the ordering of velfj, velfjo, velgj, velgjo, ufj,
****** ugj, voidfj, voidgj, qualaj, rhofj, and rhogj must be
****** maintained since vfinl assumes this order.

velfj	liquid velocity.
velfjo	liquid velocity previous time step.

velgj	vapor velocity.
velgjo	vapor velocity previous time step.
ufj	junction liquid specific internal energy.
ugj	junction vapor specific internal energy.
voidfj	junction liquid void fraction.
voidgj	junction vapor void fraction.
qualaj	junction noncondensible quality.
rhofj	junction liquid density.
rhogj	junction vapor density.
velfjs	intermediate liquid velocity used when have bad donoring.
velgjs	intermediate vapor velocity used when have bad donoring.
fjunf	form loss coefficient for area changes, foward.
fjunr	form loss coefficient for area changes, reverse.
formfj	liquid form loss term.
formgj	vapor form loss term.
mflowj	mass flow rate.
faaj	virtual mass.
fij	interphase friction.
fijo	interphase friction previous time step.
jcatn	density correction factor (sqrt of rhot/rhoj) applied to the junction convective term in choking.
jacto	density correction factor applied to the junction convective term in choking previous time step.
qualnj(1)	first noncondensible junction mass fraction.
qualnj(2)	second noncondensible junction mass fraction.
qualnj(3)	third noncondensible junction mass fraction.
qualnj(4)	fourth noncondensible junction mass fraction.
qualnj(5)	fifth noncondensible junction mass fraction.
ij1nx	from volume index.
ij2nx	to volume index.
jcnx1	index to scratch space for “from” volume. next word is same for “to” volume.
jcnx2	index to diagonal matrix element for “from” volume. next word is same for “to” volume.
jcnx3	index to off-diagonal matrix element for “from” volume. next word is same for “to” volume.
jcnxd(1)	diagonal index for sum momentum equation.
jcnxd(2)	diagonal index for difference momentum equation.
jcnxs	index to scratch space for junction.
junno	junction number for output editing.
jdissc	subcooled discharge coefficient.
jdistp	two phase discharge coefficient.
jcex	unused (bit pos. 1); ccfl flag (bit pos. 2); input ccfl flag (bit pos. 3); junction flow regime number (bit pos. 4-9).
betacc	form of ccfl correlation (0 = wallis, 1 = kutateladze).
constc	gas intercept for ccfl correlation.
constm	slope for ccfl correlation.
c0j	junction distribution coefficient.
c0jo	junction distribution coefficient previous time step.
ftgj	junction vapor interfacial shear coefficient.
ftgjo	junction vapor interfacial shear coefficient previous time step.
ftfj	junction liquid interfacial shear coefficient.

ftfjo	junction liquid interfacial shear coefficient previous time step.
xej	junction equilibrium quality based on extrapolated pressure & internal energy from jchoke.
sonicj	junction sound speed divided by the junction density ratio (jcatn).
vodfjo	junction liquid void fraction previous timestep.
vodgjo	junction vapour void fraction previous timestep.
vdfjoo	junction liquid void fraction previous timestep but one.
vdgjoo	junction vapour void fraction previous timestep but one.
fxj	wall friction interpolating factor.
fxjo	wall friction interpolation factor previous time step.
qualjo	static quality based on convected junction properties at the previous time step.
qualj	static quality based on convected junction properties.
vgjj	vapor drift velocity.
florgj	junction flow regime number in real format.
iregj	vertical bubbly/slug flow junction flow regime number in real format.
voidj	junction vapor void fraction used in the interphase drag.
jdissh	superheated discharge coefficient.
ijflg	junction direction flag (0 = 1D/1D or 1D/3D or 3D/1D, 1 = 3D/3D direction 1, 2 = 3D/3D direction 2, 3 = 3D/3D direction 3).

B.18 Comdeck lcntrlc

vlpndx	used to hold volume number and position data during hydrodynamic system sorting; holds component indexes to allow printing of component, volume, and junction information in numerical order.

B.19 Comdeck lpdatc

lpskp	system skip factor.
nloops	number of hydrodynamic systems.
lic	offset(input)/index(transient) to first hydrodynamic component in a system.
licn	number of hydrodynamic components in a system.
liv	offset(input)/index(transient) to first volume in a system.
livn	number of volumes in a system.
livnn	the number of the first volume in a system.
lij	offset(input)/index(transient) to first junction in a system.
lijn	number of junctions in a system.
lsuces	for each system: 0 if no need to repeat advancement with reduced time step, 1 if excessive truncation error, 2 if water property error, 3 if non-diagonal matrix.
lpackr	packing flag from packer subroutine.
llvect	index to list vector for system.
lnoncn	number of noncondensible gasses in system.
lnonmf	number of molten metal species in system.
nnz	number of nonzero elements in solution matrix.
nnz2	number of matrix elements allowed during solution.
nvr	order of matrix.
nvrp	order of matrix plus one.
ixip	pointer to control array for matrix inversion routine.
ixipr	pointer indicating first nonzero in row of solution matrix.

ixirn	similar function to ixirnr for previously inverted array.
ixirnr	index to array holding position of nonzeros for matrix.
ixivrn	pointer to array holding indexes of matrix elements for nearly implicit advancement.
ixipx	pointer to ip array during pminvd call.
ixirnx	pointer to irn array during pminvd call.
ixsopr	index to right hand side and solution results.
ixcofp	index to array holding nonzero values of matrix.
ixw	index to work array for matrix inversion routine.
sflag	indicates whether new scaling and solution order needed.
lsysnm	system name.
systms	system mass from sum of densities times volumes.
systmc	system mass from conservation law applied to system, new time value.
systmo	same as systmc but old time value.
sysmer	system mass error.
sysebt	system error measure.
sysdtc	system courant limit.
gerrs	error at which scaling and solution order will be re-evaluated.

The quantities, nnz, nnz2, nvr, nvrp, ixip, ixipr, ixirn and sflag, use two loccation, one for volume oriented matrices, the second for junction oriented matrices.

B.20 Comdeck lvectrc

lvaccm	pointer to accumulator list.
lvpump	pointer to pump list.
lvsepr	pointer to sepalrator list.
lvvalv	pointer to valve list.
lvturb	pointer to turbine list.
lvjtmx	pointer to jetmixer list.
lvrvol	pointer to real volume list.
lvtvol	pointer to time dependent volume list.
lvjusr	pointer to junction-source list.
lv3d	pointer to multidimensional components.
lvprz	pointer to pressurizer component junction list.
lvptr	list values.

B.21 Comdeck maxmemc

maxscm	maximum amount of primary memory.
maxlcm	maximum amount of secondary memory (none now used).

B.22 Comdeck mtblc

nmtbls	number of materials.
mtbptr	offset to material data.
mtbnum	composition or material number.
mtblen	length of data for material.
mtbl	used for integer information in table.

mtblr	used for real (floating point) information in table.

B.23 Comdeck mxnfcdc

mxnfld	maximum number of fluids available for use.
wmoles	molecular weight of fluid.
tpfnam	names of thermodynamic properties files from which steam lestabare to be obtained for needed fluids; file names initialized in blkdta, and overridden by file names given in tpfncl and tpfnin.
tpfncl	names of thermodynamic properties files as obtained from command line.
tpfnin	names of thermodynamic properties files as obtained from 120-129 input cards; overridden by file names given in tpfncl.
fsymbl	symbols for available fluids; set in blkdta.

B.24 Comdeck parmc

alhir	inner radius of lower head in vessel.
amult	a multiplier for some storage space requirements; set to 2.0 in subr. step.
atrhgt	height of debris used for ATR configuration, saved in mupdat from one call to the next.
cnpool	thermal conductivity of molten pool (w/m.k).
coni	used in mesh generation; always 0.0.
conj	used in mesh generation; always 0.0.
contr	convergence parameter in COUPLE numerical solution.
depth	depth of localized molten pool in debris bed for use as characteristic length in rayleigh number.
depthp	depth of plane(thickness) for case of plane geometry.
dh	thickness of lower head of vessel.
dradin	inner radius of slumped material(axisymmetric geometry) or depth of plane(plane geometry).
dradis	outer radius of region in mesh that can fill with debris.
dtcpl	current COUPLE time step.
dtold	last COUPLE time step.
emissm	emissivity for internal radiation in material with id 1.
ftempc	fixed input temperature for tempsets.
hed1	first COUPLE title line.
hed2	second COUPLE title line.
height	height of debris bed, including dh.
radc	input radiation temperature.
rlxn	relaxation parameter in COUPLE numerical solution.
rlxo	1.0 - rlxn.
rtemp	input radiation reference temperature.
seti	used in mesh generation; always 0.0.
setj	used in mesh generation; always 0.0.
sigf	input radiation heat trans. coeff.
tempin	initial temperature of COUPLE mesh.
time	problem time, same as timehy.
rap	rayleigh number for debris bed molten pool.

tpool	average temperature for debris bed molten pool.
tvol	total volume of debris.
uncon	multiplier for dimensions that are input.
zpbot	bottom elevation of molten pool (m).
zptop	top elevation of molten pool (zpbot + depth) (m).

B.25 Comdeck przdatc

przlvl	liquid level in pressurizer.
przlln	pressurizer liquid level node index.
srgjun	surge line junction index.
pzhtcf	user supplied interface heat transfer coefficient for liquid.
pzhtcg	user supplied interface heat transfer coefficient for vapor.
przndS	number of pressurizer volumes (and junctions including the surge line junction).
przjnx	pressurizer junction indices starting from the top.
przvnx	pressurizer volume indices starting from the top.

B.26 Comdeck radhtcc

dynamic file for radiation heat transfer is file 38.

definitions for radiation heat transfer:

nrset	number of sets of radiation surfaces.
irhflx	offset to radiation heat fluxes which are for all heat str.

-------above two words are 1st and second words in the file----

nrsskp,nrhskp are parameters

nrsskp	skip factor for radiation ht set variables.
nrhskp	skip factor for radiation ht surface variables.
	- the following block is repeated for each set of radiating surfaces.
	- these blocks arranged sequentially in order of increasing set number.
setno	the set number. may be negative during input processing.
refset	the set number from which this set gets its view factors.
refset	must always be less than setno.
nrh	number of radiation surfaces in the set.
trmin	minimum surface temperature of surfaces. if all surfaces have temp less than trmin no radiation calculated for the set.
voidmn	minimum vapor void fraction. if boundary volume void fraction is less than voidmn no radiation calculated for the set.
timron	last time radiation ht calculation began for a set.
timron	positive means radiation calc is off.
timron	negative means radiation calc is on.
timrof	last time radiation ht ended for a set.
irhoff	offset to radiation ht surface data for a set.
ivewof	offset to vfij matrix for a set.

the following variables are repeated for each radiation surface:

jrh	heat conductor surface number (input),
jlr 0	left surface (input),
1	right surface (input).

in iradht routine jlr is changed to the sequence number of radition surface so that it points to item of location in the heat structure dynamic file. this index is:

-	for left surface,
+	for right surface,
emis	emissivity of a surface,
itemof	offset to surface temperature (pointer into heat str file),
ias	offset to surface area (pointer into heat str file),
itg	offset to fluid temperature (pointer into volume file),
qrad	radiation heat flux for a surface.

the following is set of view factor matrices. there may be one nrh by nrh view factor matrix for each set of radiating surfaces, but there may be fewer since one set can refer to another set for its view factors. the first nrh elements in each matrix contains the view factors from the 1st surface in the set to the nrh surfaces, etc.

vfij view factors for first set (nrh(1)**2 words).
vfij view factors for next set..etc.

the following two words are repeated nhtstr times, where nhtstr is the total number of heat structures in the problem (as stored in the heat structure file).

qlrad	left radiation heat flux to heat conductors. (not just radiative heat conductors).
qrrad	right radiation heat flux to heat conductors. (not just radiative heat conductors).

B.27 Comdeck rkincc

****
Point kinetics power option
****

rknum	number of reactor kinetics equations.
rknumd	number of delay groups.
rkopt	control for calculating cn functions (bits 1 and 2); number of table coordinates (4, 8, and 16 bits); separable/table feedback flag (32 bit), initialization flag (64 bit), point/one-d (flag 128 bit), unable to initialize flag (256 bit).
rkoffd	offset for loop over delay groups (point). offset/index to kinetics mesh information (one-d).
rkoffa	offset for loop over all kinetics equations (point), last filndx(21) value to compute indexes.
rknsc	number of scram data entries.
rksptr	pointer to scram data.

rkdnpt	pointer to density reactivity data (point separablefeedback), pointer to coordinate data (tabular feedback), offset/index to composition data (oned).
rkdppt	pointer to doppler reactivity data (point separable feedback), pointer to tabular data (point tabular feedback), offset/index to zone data (one-d).
rknvfb(1)	number of volumes in reactivity feedback.
rknvfb(2)	pointer to volume feedback data.
rknsfb(1)	number of heat structures in reactivity feedback.
rknsfb(2)	pointer to heat structure feedback data.
rkbol	delayed neutron fraction over generation time.
rkro	reactivity bias after initialzation.
rkrn	current reactivity.
rkomeg	current reciprical period.
rkslob	source divided by delayed neutron fraction over generation.
rkdt	reactor kinetic time step.
rksum	holds delayed neutron summation.
rkqval	fraction of fission energy released at fission.
rkfu38	u239 production factor.
rkpow	total reactor power, i.e., the sum of fission power and gamma decay power.
rkpowa	actinide decay power.
rkpowf	power from fission.
rkpowg	power from gamma decay, i.e. the sum of fission product decay power and actinide decay power.
rkpowk	fission product decay power.
rkdepv	dependent variables being advanced in time.
rkfi	delayed neutron and gamma yield fractions.
rklmda	decay constants for delayed neutrons and gamma decay.
rkcnh1	
rkcnh2	
rkcnh3	integration weights at half interval.
rkcnf1	
rkcnf2	
rkcnf3	integration weights at full interval.
rknscr(1)	general table number or control variable number.
rknscr(2)	offset or index to general table or control variable.
rkdeni(1)	number of quantities per set in density table.
rkdeni(2)	number of quantities in table.
rkdeni(3)	current position in table.
rkdenr	density reactivity table if separable, coordinates if table.
rkdopi(1, 2, 3)	same as for rkdeni except for doppler table.
rkdopr	doppler reactivity table.
rkvoln(1)	volume number.
rkvoln(2)	offset or index to volume data.
rkvwf	density weight for volume feedback.
rkvta	volume temperature reactivity coeeficient.
rkhtno(1)	heat structure number.
rkhtno(2)	offset to heat structure data.
rkfwf	doppler weight for heat structure feedback.
rkfta	heat structure temperature reactivity coeeficient.

rk	ltab table data.
rkfdcd	code to determine if new interpolating elements are needed.
rknden	coordinate subscripts.
rkcoef	interpolating elements.

****

One dimensional kinetics power option

****

rkdt	kinetic ime step advancement interval.
rkfu38	number of u238 atoms produced per fission.
rknum	number of kinetics equations.
rknumd	number of delayed neutron groups.
rknumg	number of gamma/actinide decay heat equations.
rkoa	coefficients for cross sections.
rkoalp	averaged void fractiona in volume region.
rkob	averaged poison density in volume region.
rkobcf	bottom boundary condition flag.
rkobtb	bias thermal buckling.
rkocal	active length of control rod.
rkocc	convergence criterion.
rkocfr	control fraction for each axial mesh.
rkocfn	number of control rod groups for computation of control fraction on each axial mesh.
rkoci	insertion depth of control rod group.
rkocid	material composition identifier.
rkocn	number of compositions.
rkocrc(1)	identifier of control block(or general table) moving control rod group.
rkocrc(2)	offset/index in control component data base for data.
rkocrn(1)	control rod group identifier
rkocrn(2)	offset/index in control rod database for data for this control rod group.
rkocrf	flag which identifies which end of the reactor the control rods enter.
rkocri	control rod group identifier.
rkocs	cross section arrays.
rkocwf	control rod group weight factor.
rkocxa	reactor cross sectional area.
rkoden	average fluid density in volume region.
rkodpc	delayed neutron precursor inventories.
rkodpv	decay heat precursor densities for each zone.
rkoef	extrapolation factor.
rkoefp	fission power normalization factor.
rkoegv	eigenvalue.
rkoesf	eigenvalue shift factor.
rkofbm(1)	composition number for kinetics mesh interval.
rkofbm(2)	offset/index to composition data.
rkofbs	number of heat structures in heat structure region.
rkofbv	number of volumes in volume region.
rkofi	decay heat group yields.
rkoiop	integration option.
rkolmd	decay heat decay constants.
rkomfc	maximum fractional flux change parameter.
rkomnd	height of top of each of kinetic mesh interval from bottom of reactor core.
rkomxd	kinetic matrix diagonal terms.

rkomxl	kinetic matrix lower triangular terms.
rkomxu	kinetic matrix upper triangular terms.
rkonc	number of material compositions
rkoncr	number of control rod groups.
rkonn(1)	number of kinetic mesh intervals.
rkonn(2)	offset/index to output k-infinity data(rkorki).
rkonn(3)	offset/index to nodal fluxes(rkophi).
rkonn(4)	offset/index to scratch vector sor1 in steady state(rkosr1) offset/index to nodal powers in transient(rkop1).
rkonn(5)	offset/index to nodal relative powers in transient(rkop2).
rkonn(6)	offset/index to scratch vector sor2 in steady state(rkosr2) offset/index to energies in transient(rkop3).
rkonn(7)	offset/index to relative energies in transient(rkop4).
rkonn(8)	offset/index to nodal delayed precursor inventories(rkodpc).
rkonn(9)	offset/index to prompt omega data(rkoomp).
rkonn(10)	offset/index to delayed omega data(rkoomd).
rkonn(11)	offset/index to scratch for nodal sources(steady state-rkos transient-rkosr1).
rkonn(12)	offset/index to below diagonal kinetic matrix elements(rkom).
rkonn(13)	offset/index to diagonal kinetic matrix elements(rkomxd).
rkonn(14)	offset/index to above diagonal kinetic matrix elements(rkomxu).
rkonn(15)	offest/insex to bias thermal buckling array(rkobtb).
rkonn(16)	offset/index to target relative power distribution(rkotpd).
rkonn(17)	offset/index to zone average property data.
rkonn(18)	offset/index to composition data.
rkonn(19)	offset/index to control rod insertion data.
rkonn(20)	offset/index to control fraction data.
rkonn(21)	offset/index to zone power data.
rkonn(22)	offset/index to kinetics mesh data.
rkonn(23)	offset/index to next free location after kinetics data.
rkonsr	number of heat structure regions in a zone.
rkonvr	number of volume regions in a zone.
rkonz	number of zones.
rkoomd	delayed omega data.
rkoomp	prompt omega data.
rkop1	kinetic mesh fission power.
rkop2	kinetic fission power distribution.
rkop3	kinetic fission energy deposition.
rkop4	kinetic fission energy deposition distribution.
rkophi	neutron flux.
rkopow	zone total power.
rkopt	reactor power control bits
rkopwa	zone actinide decay power.
rkopwf	zone fission power.
rkopwg	zone gamma power.
rkopwk	zone fission product decay power.
rkorid	control rod insertion depth.
rkorki	K-infinity data.
rkosn(1)	heat structure identifier.
rkosn(2)	offset/index in heat structure database for data for this heat structure.
rkosr1	scratch vector in steady state, nodal sources in transient.

rkosr2	scratch vector in steady state, not used in transient.
rkosr3	nodal sources in steady state, not used in transient.
rkossa	steady state source acceleration flag.
rkossf	steady state selection flag.
rkoswf	heat structure weight factor
rkotcf	top boundary condition flag.
rkotdp	time difference parameter.
rkotev	target eigenvalue.
rkotf	structure region average fuel temperature.
rkotm	volume region average moderator temperature.
rkotpd	target axial power distribution.
rkotru	reference termperature units flag.
rkotsc	time step control.
rkovn(1)	volume identifier.
rkovn(2)	offset/index in volume database for data for this volume.
rkovwf	volume weight factors, (1) factor for average void fraction or density, (2) factor for average fluid temperature, and 3) factor for average poison density.
rkozid	zone identifier.
rkozn(1)	zone identifier for kinetics mesh interval.
rkozn(2)	offset/index to zone average property data.
rkpow	total reactor power.
rkpowa	total actinide decay power.
rkpowf	total fission power.
rkpowg	total gamma decay power.
rkpowk	total fission product decay power.
rkqval	total energy per fission.

B.28 Comdeck scddatc

maxpp	maximum number of points in nheati created power/time arrays.
maxpz	maximum number of axial profile arrays.
ndax	maximum number of axial nodes.
ndcomp	maximum number of components.
ndgrid	maximum number of grid spacers.
ndmatr	maximum number of materials in component.
ndrd	maximum number of radial nodes.
ndrg	maximum number of regions.
ndtime	maximum number of different times steps.
nxdbrg	maximum number of debris regions.
icomp	current component.

B.29 Comdeck separc

vover	void limit for ideal vapor outflow.
vunder	void limit for ideal liquid fall back.

B.30 Comdeck sscntrc

l24skp	skip factor for steady state block. note...packed as...(30 bit) = total count, (30 bit) = edit count
smxdrh	count of max d(rho*h)/dt for a volume.
smxrho	count of max (rho-rhom) for a volume.
smndrh	count of min d(rho*h)/dt for a volume.
smnrho	count of min (rho-rhom) for a volume.
uo	old time mixture u for a volume.

B.31 Comdeck ssiblkc

ssiskp	skip factor, the number of words from compid to cmpiii.
npumps	no. of pump component, controller pairs.
nstctl	no. of steam component, controller pairs.
nfeeds	no. of feed component, controller pairs.
compid	component no. of regulated component.
contid	controller no. regulating compid.
cvrtno	controller type no.
cvrtnm	controller type name.
cmptno	component type no.
cmptnm	component type name.
cmpvnm	component search or control variable name.
cmpvno	component search or control variable no.
cmptrp	component controlling trip no.
cmpxxx	dummy real variable.
cmpiii	time dependent data table type.
pcmpid	component no. of pressurizer component.
pcnnec	component no. of pressurizer connection.
pcntno	connection component type no.
pcntnm	connection component type name.
pcmtno	pressurizer component type no.
pcmtnm	pressurizer component type name.
pcmvnm	pressurizer component search variable name.
pcmvno	pressurizer component search variable number.
pcmtrp	pressurizer component trip no.
pzpres	pressurizer pressure.
pzityp	pressurizer component time dependent table type no.

total file length = 4 + ssiskp*(npumps + nstctl + nfeeds) + ssiskp (if card 147 is input)
since the max value of npumps, nstctl, and nfeeds is 6 each,
then the largest file size for this file is 213 words.

B.32 Comdeck statcc

block	for statistics during advancement.
strdc1	number of repeated time steps in entire problem.
strdc2	number of repeated time steps in this major print interval.

stsjpt	pointer to junction array.
stsatp	total number of advancements.
stsreq	total number of requested advancements.
stscpu	cpu time required.
stsdtn	minimum time step during edit.
stsdtx	maximum time step during edit.
stsdta	sum of time steps for averge over edit.
stlte1	number of times volume had largest mass error in entire problem.
stlte2	number of times volume had largest mass error in this major print interval.
strxl1	number of times quality adjustment in volume caused reduced time steps in entire problem.
strxl2	number of times quality adjustment in volume caused reduced time steps in this major print interval.
strex1	number of times state extrapolation in volume caused reduced time steps in entire problem.
strex2	number of times state extrapolation in volume caused reduced time steps in this major print interval.
strte1	number of times mass error in volume caused reduced time steps in entire problem.
strte2	number of times mass error in volume caused reduced time steps during this major print interval.
strpe1	number of times water property error in volume caused reduced time steps in entire problem.
strpe2	number of times water property error in volume caused reduced time steps during this major print interval.
stscl1	number of times volume had smallest courant limit in entire problem.
stscl2	number of times volume had smallest courant limit in this major print interval.
strcl1	number of times courant limit for this volume caused reduced time step in entire problem.
strcl2	number of times courant limit for this volume caused reduced time step in this major print interval.
stjck1	number of times junction choked in entire problem.
stjck2	number of times junction choked in this major print interval.
stccf1	number of times junction used ccfl correlation in entire problem.
stccf2	number of times junction used ccfl correlation in this major print interval.

B.33 Comdeck statecc

wmolea	molecular mass of non-condensible gas.
rax	gas constant of non-condensible gas.
dcvax	same as above.
cvaox	same as above.
uaox	term in u = uao + integral (cv*dt) where u is internal energy.
tao term in cv	cvao + dcva*(t - tao) where cv is heat capacity and t is temperature.
noncn	number of non-condensible gasses.
nonmf	number of molten metal species when mmfld option is defined; also extra integer to maintain 8 byte boundary for 32 bit machines when mmfld is not defined.
prop	array for sth2x calls, also used for scratch.
s	same as above.

B.34 Comdeck stcblkc

nfluid	fluid number:
1	light water,
2	heavy water,
3	hydrogen,
4	lithium,
5	potassium,
6	helium,
7	nitrogen,
8	sodium,
9	NaK,
10	lithium-lead,
11	ammonia.
ndxstd	pointer in fa array to first word of steam tables for fluid with fluid number nfluid.

B.35 Comdeck stcomc

lstcom	number of words (single precision) in the /stcom/ common block.
ttrip	triple point temperature of current fluid.
ptrip	triple point pressure of current fluid.
vtrip	triple point volume of current fluid.
tcrit	critical point temperature of current fluid.
pcrit	critical point pressure of current fluid.
vcrit	critical point volume of current fluid.
tmin	minimum allowed temperature for current fluid.
pmin	minimum allowed pressure for current fluid.
tmax	maximum allowed temperature for current fluid.
pmax	maximum allowed pressure for current fluid.
nt	number of temperatures in steam tables for current fluid.
np	number of pressures in steam tables for current fluid.
nst	number of saturation temperatures in steam tables for current fluid.
nsp	number of saturation pressures in steam tables for current fluid.
it3bp	base pointer to third table in steam tables for current fluid.
it4bp	base pointer to fourth table in steam tables for current fluid.
it5bp	base pointer to fifth table in steam tables for current fluid.
nprpnt	number of thermodynamic properties in steam tables times nt for the current fluid.
it3p0	pointer to zeroth word of third table in steam tables for current fluid.

B.36 Comdeck trnhlpc

ixipr	pointer indicating first nonzero in row of solution matrix.
ixip	pointer to control array for matrix inversion routine.
ixw	pointer to work array for matrix inversion routine.
ixpv	pointer to array for right hand side.
ixirnr	pointer to array holding position of nonzeros for matrix inversion routine.
ixirn	similar function to ixirnr for previously inverted array.

ixcofp	pointer to array holding nonzero values of matrix.
nvr	order of matrix, also number of non-time dependent volumes.
nnz	number of nonzero elements in solution matrix.
gerr,gerrs	estimated error in inversion process, and error at which. scaling and solution order will be reevaluated.
mtype	type of matrix multiply, needed in call to matrix product routine.
sflag	indicates whether new scaling and solution order needed.
lhtsol	used to determine scratch storage lengths in trnset.

B.37 Comdeck trpblkc

ntvskp	variable trip skip factor.
ntlskp	logical trip skip factor.
ntrpnv	number of variable trips.
ntrpnl	number of logical trips.
ntrpof	offset to logical trips.
ntrps1(1)	first trip number to terminate advancement.
ntrps1(2)	offset to trip during input processing, index to trip during transient.
ntrps2(1 and 2)	same as ntrps1 except for second trip.
trptim	time trip was set, also used as trip switch. negative if trip not set or false, positive if set or true and the value is the time the trip was set.
trptm	same variable as trptim but used with different subcripts. trptim used in loops over all trips, trptm used when subscripting obtained from itrscn subroutine is used.
ntrpno	trip number.
ntrpop	initialization flag (1 bit), latch code (2 bit), timeof flag left variable (4 bit), timeof flag right variable (8 bit), sign bit on left trip number (16 bit), sign bit on right trip number (32 bit), operation code (shifted left 24 bits).
ntrcv1	variable code, left side.
ntrnv1	number part of variable code, left side.
ntrpc1(1)	block number of left variable.
ntrpc1(2)	offset to left variable during input, index to left variable during transient.
ntrcv2 ntrnv2	
ntrpc2	same as ntrcv1, ntrnv1, and ntrpc1 but for right variable.
trpcon	constant on right side.
ntrtr1(1)	left operand trip number.
ntrtr1(2)	offset to left trip time during input processing, index to left trip time during transient.
ntrtr2(1) ntrtr2(2)	are the same as ntrtr1(1) and ntrtr1(2) except for right operand.

B.38 Comdeck tsctlcc

time step control limits.
see dtstep for common statement.

errhi	upper limit on mass error control.
errlo	lower limit on mass error control.

B.39 Comdeck tstpctc

curclm	limit value for card pointer.
curctl	pointer to current card.
curcmi	current counter for plot and minor edit.
curcmj	current counter for major edit.
curcrs	current counter for restart.
tspend	time end for associated time step values and counters.
dtmin	miminum time step.
dtmax	maximum time step.
tsppac	advancement control.
tsppmi	minor edit frequency.
tsppmj	major edit frequency.
tspprs	restart frequency.

B.40 Comdeck turbinc

turscm	shaft component number.
turctr(1)	turbine disconnect trip number.
turctr(2)	offset for above during input, index during transient.
turupj	junction number upstream of stage.
turvel	turbine rotational velocity.
turint	turbine moment of inertia.
turfr	turbine friction factor.
turpow	power developed by turbine.
turtrq	torque developed by turbine.
tureff	turbine efficiency.
turrds	mean stage blade radius.
turdef	max. stage efficiency.
turx	stage reaction ratio.

B.41 Comdeck ufilesc

input	input file.
output	printed output file.
rstplt	restart-plot file.
stripf	strip file.
plotfl	scratch file for internal plot capability.
sth2xt	used for all thermodynamic property files.
tty	online screen file.
jbinfo	user file to be copied to job output file.
filsch	holds file names used in open statements (except for thermodynamic properties files); default values set by data statements in blkdta; user supplied values can be optionally entered in some machine versions.

B.42 Comdeck voldatc

ivskp	volume skip factor.
nvols	number of volumes.
vctrl	time dependent volume flag (bit pos. 1); equilibrium flag (bit pos. 2); wall friction flag (bit pos. 3); input flag (bit pos. 4); vapor disappearance flag (bit pos. 5); accumulator flag (bit pos. 6); pump flag (bit pos. 7); input water packer flag (bit pos. 8); new status flags, initialization type during input (bit pos. 9-19); old status flags (bit pos. 20-30); input bundle flag (bit pos. 31).
volmat	fluid type in volume.
volno	volume number for editing.
imap	map, regime, and flags. Three quantities, one per coordinate. Flow regime map infomation (bit pos. 1-6); horizontal stratification flags (bit pos. 7-9); input vertical stratification flag (bit pos. 10); vertical stratification flags (bit pos. 11-12); experimental friction being used (bit pos. 13); unused (bit pos. 14); flag that oordinate direction 1 is being used (bit pos. 15); unused (bit pos. 16-18); flow regime number (bit pos. 19-24); metal appearance flags (bit pos. 25-26); laminar friction factor, 64 if 0, 96 if 1 (bit pos 27); unused (bit pos. 28-29); reflood flag (bit pos. 30).
v	volume.
recipv	reciprocal of volume (v), zero if v is zero.
avol	area of volume, three quantities, one per coordinate.
dl(1)	volume length, three quantities, one per coordinate.
diamv	equivalent flow diameter, three quantities, one per coordinate.
roughv	wall roughness factor for direction 1. As input reset in icmpn1 to colebrook full turb friction fac.
recrit	critical reynolds number, three quantities, one per coordinate. Fric fac = const; see roughv.
p	average pressure.
po	average pressure previous time step.
uf	liquid specific internal energy.
ufo	liquid specific internal energy previous time step.
ug	vapor specific internal energy.
ugo	vapor specific internal energy previous time step.
voidf	liquid void fraction.
voidg	vapor void fraction.
voidgo	vapor void fraction previous time step.
quala	noncondensible quality.
qualao	noncondensible quality previous time step.
boron	boron density (mass of boron per cell volume).
borono	boron density previous time step.
quals	static quality.
quale	equilibrium quality.
rho	total density.

rhom	total density for mass error check.
rhoo	total density previous time step.
****** warning: the ordering of rhof and rhog must be maintained	
****** since fidis assumes this order.	
rhof	liquid density.
rhog	vapor density.
satt	saturation temperature based on the steam partial pressure.
temp	used in r level subroutines and is usually the same as satt.
tempf	liquid temperature.
tempg	vapor temperature.
velf(1)	average liquid velocity in a volume, three quantities, one per coordinate.
velg	average vapor velocity in a volume, three quantities, one per coordinate.
sounde	homogeneous equilibrium sound speed. also, used for scratch in hydro.
dsnddp	partial derivative of sounde w/r to pressure. also, used for scratch in hydro.
sathf	liquid specific enthalpy at saturation conditions.
sathg	vapor specific enthalpy at saturation conditions.
betaff	liquid isobaric coefficient of thermal expansion at bulk conditions. Also, used for scratch in hydro.
betagg	vapor isobaric coefficient of thermal expansion at bulk conditions. Also, used for scratch in hydro.
csubpf	liquid specific heat capacity at constant pressure at bulk conditions. Also, used for scratch in hydro.
csubpg	vapor specific heat capacity at constant pressure at bulk conditions. Also, used for scratch in hydro.
viscf	liquid viscosity. also, used for scratch in hydro.
viscg	vapor viscosity. also, used for scratch in hydro.
sigma	surface tension. also, used for scratch in hydro.
thconf	liquid thermal conductivity.also, used for scratch in hydro.
thcong	vapor thermal conductivity. also, used for scratch in hydro.
pps	vapor partial pressure.
dotm	vapor generation rate per unit volume.
dotmo	vapor generation rate per unit volume previous time step.
hif	liquid side interfacial heat transfer coefficient per unit volume.
hig	vapor side interfacial heat transfer coefficient per unit volume.
gammaw	vapor generation rate at the wall per unit volume.
q	total heat transfer rate from wall to fluid.
qwg	heat transfer rate from wall to vapor.
drfdp	partial derivative of rhof w/r to pressure.
drfduf	partial derivative of rhof w/r to liquid specific internal energy.
drgdp	partial derivative of rhog w/r to pressure.
drgdug	partial derivative of rhog w/r to vapor specific internal energy.
drgdxa	partial derivative of rhog w/r to noncondensible quality.
dtfdp	partial derivative of tempf w/r to pressure.
dtfduf	partial derivative of tempf w/r to liquid specific internal energy.
dtgdp	partial derivative of tempg w/r to pressure.
dtgdug	partial derivative of tempg w/r to vapor specific internal energy.
dtgdxa	partial derivative of tempg w/r to noncondensible quality.
dtdp	partial derivative of satt w/r to pressure.
dtdug	partial derivative of satt w/r to vapor specific internal energy.
dtdxa	partial derivative of satt w/r to noncondensible quality.

floreg	flow regime number in real format, three quantities, one per coordinate.
htcoff	heat transfer coefficient between slab and fluid-liquid.
htcofg	heat transfer coefficient between slab and fluid-vapor.
hifo	liquid side interfacial heat transfer coefficient per unit
volume	previous timestep.
higo	vapor side interfacial heat transfer coefficient per unit.
volume	previous timestep.
qualan	noncondensible mass fraction, five quantities, one per species.
gaman	noncondensible generation rate per unit volume, five quantities, one per species.
enthn	enthalpy of noncondensible source, five quantities, one per species.
gamas	solute generation rate per unit volume.
enths	enthalpy of the solute source.
vo	volume previous time step.
qualno	noncondensible mass fraction previous time step, five quantities, one per species.
rhogo	vapor density previous time step.
ppso	vapor partial pressure (old-time).
ustm	vapor specific internal energy at pps and tempg with non-condensable present.
ustmo	vapor specific internal energy at pps and tempg with non-condensablepresent (old-time).
ggas	cell centered gas mass flux, three quantities, one per coordinate.
gliq	cell centered liquid mass flux, three quantities, one per coordinate.
velfo	volume average liquid velocity previous timestep, three quantities, one per coordinate.
velgo	volume average vapor velocity previous timestep, three quantities, one per coordinate.
fstrt	horizontal stratification interpolating factor, three quantities, one per coordinate.
fwalf	liquid wall friction coefficient, three quantities, one per coordinate.
fwalg	vapor wall friction coefficient, three quantities, one per coordinate.
vctrln	position of volume in volume block.
vctrld	index to diagonal matrix element.
vctrls	index to volume scratch space.
sth2xv	index data for sth2x water property subroutines.
invfnd	index to inverted junction table.
sinb	sine function of volume vertical angle, three quantities, one per coordinate.
hvmix	volume mixture enthalpy.
ptans	pitch between fuel plates (ans).
span	length of fuel plates (ans).
pecltv	volume peclet number, three quantities, one per coordinate.
tsatt	saturation temperature based on the total pressure.
fshape	wall friction shape factor (one per coordinate).
fmurex	viscosity ratio for wall friction (one per coordinate).
frica	constant term in experimental friction correlation (one per coordinate).
fricb	multiplier term in experimental friction correlation (one per coordinate).
fricc	power term in experimental friction correlation (one per coordinate).
invhtf	index to inverted heat structure table.
hydxc(1)	change along inertial x axis due to moving from face 1 to center of volume along local x coordinate.
hydxc(2)	change along inertial x axis due to moving from center of volume to face 2 along local x coordinate.
hydxc(3)	change along inertial x axis due to moving from face 3 to center of volume along local y coordinate.

hydxc(4)	change along inertial x axis due to moving from center of volume to face 4 along local y coordinate.
hydxc(5)	change along inertial x axis due to moving from face 5 to center of volume along local z coordinate.
hydxc(6)	change along inertial x axis due to moving from center of volume to face 6 along local z coordinate.
hydyc(1)	change along inertial y axis due to moving from face 1 to center of volume along local x coordinate.
hydyc(2)	change along inertial y axis due to moving from center of volume to face 2 along local x coordinate.
hydyc(3)	change along inertial y axis due to moving from face 3 to center of volume along local y coordinate.
hydyc(4)	change along inertial y axis due to moving from center of volume to face 4 along local y coordinate.
hydyc(5)	change along inertial y axis due to moving from face 5 to center of volume along local z coordinate.
hydyc(6)	change along inertial y axis due to moving from center of volume to face 6 along local z coordinate.
hydzc(1)	change along inertial z axis due to moving from face 1 to center of volume along local x coordinate.
hydzc(2)	change along inertial z axis due to moving from center of volume to face 2 along local x coordinate.
hydzc(3)	change along inertial z axis due to moving from face 3 to center of volume along local y coordinate.
hydzc(4)	change along inertial z axis due to moving from center of volume to face 4 along local y coordinate.
hydzc(5)	change along inertial z axis due to moving from face 5 to center of volume along local z coordinate.
hydzc(6)	change along inertial z axis due to moving from center of volume to face 6 along local z coordinate.
hyposv(1)	coordinate along x inertial axis of vector from center of rotation to center of volume.
hyposv(2)	coordinate along y inertial axis of vector from center of rotation to center of volume.
hyposv(3)	coordinate along z inertial axis of vector from center of rotation to center of volume.
gravv(1)	coordinate of gravity along inertial x coordinate.
gravv(2)	coordinate of gravity along inertial y coordinate.
gravv(3)	coordinate of gravity along inertial z coordinate.
$	endif
$if	def,selap,6.
idbvol	indicator of whether l-th index of RELAP5 volume contains debris region; 0 = no, 1 = yes.
mdbvol	value for lth RELAP5 volume index of index m for referencing arrays in common block debcom.
ndbvol	value for l-th RELAP5 volume index of index n for referencing arrays in common block debcom.

Appendix C. THE INP DATA INPUT PACKAGE

The INP set of subroutines constitutes a convenient data input package for use with Fortran programs. To the user, the package offers: free form input; card number to identify the cards; automatic removal of cards containing duplicate card numbers; arbitrary ordering of input cards except for duplicate cards; arbitrary use of comment cards and comments on data cards; ease of preparing cases in which only moderate changes are made from case to case; and a listing of the card data. For the programmer, the package is a convenient method for implementing a highly user oriented data input scheme and includes: extensive checking of the amount and mode (integer, floating point, or alphanumeric) of data; automatic expansion of sequential and overlay type of input data; deletion of unneeded cards; checking whether extraneous input has been entered; and the ability to detect several errors during input checking. The INP package also uses an output subroutine which reduces the programmer burden in formatting output pages.

C.1 User Aspects

This section describes the INP package as seen by the program user.

C.1.1 Data Deck Organization

The data deck contains input for one or more problem sets. No relationship is assumed between problem sets. Each problem set consists of one or more cases in which the input data for cases other than the first consist of the data from the previous case plus modification cards entered for the present case. Input data for cases are separated by slash cards; the final case is terminated by a period card instead of a slash card. The period card also serves as the separator between problem sets. A slash card has a (/) as the first non-blank character on a card; a period card has a (.) as the first non-blank character. Comments may follow the slash and period on slash and period cards.

A list containing a card sequence number and the card image of each card is printed at the beginning of printed output for each case. The card sequence number starts at one for each case. The first line of the list contains, “LISTING OF INPUT DATA FOR CASE n”, where n is the case number.

C.1.2 Title Card

A title card is designated by an equal sign (=) as the first non-blank character on a card. The remainder of the card can have any alphanumeric characters. The information on the title card and the current date are printed at the top of every page following the input data listing. One title card should be entered for each case. If more than one title card is entered in a case, the contents of the last title card are used for the page heading. The heading contains only the date if no title card is entered for a case.

C.1.3 Comments Cards

An asterisk (*) or a dollar sign ($) appearing as the first non-blank character identifies the card as a comments card. Any information may be entered on the remainder of the card. Blank cards are treated as comments cards. There is no processing of comment cards other than 1isting them in the card list.

C.1.4 Data Cards

All cards other than title cards, comments cards, slash cards, or period cards are considered data cards. The data cards contain a varying number of fields which may be decimal integer, decimal floating point, alphanumeric, octal, or hex. The rules for specifying fields are completely described in the CVI[2] subroutine documentation, but brief rules sufficient for most users are given here.

Blanks preceding and following fields are ignored. A decimal field is started by either a digit (O through 9), a sign (+ or -), or a decimal point (.). A comma or a blank (with one exception noted below) terminates the decimal field. The decimal field has a number part, and optionally an exponent part. A decimal field without a decimal point or an exponent is a decimal integer field; a field with either a decimal point or an exponent or both is a decimal floating point field. A decimal floating point field without a decimal point is assumed to have a decimal point immediately in front of the first digit. The exponent denotes the power of ten to be applied to the number part of the field. The exponent part is a sign, an E or D, or an E or D and a sign followed by a number giving the power often. Rules for decimal floating point numbers are identical to those for entering data in Fortran E or F formatted fields except that no blanks (one exception) are allowed between characters. Floating point data punched by Fortran programs can be read; to permit this, a blank following an E or D denoting an exponent is treated as a plus sign. Acceptable ways of entering floating point numbers are illustrated by the following six fields all containing the quantity 12.45,

12.45,+12.45 1245-2 1.245+1,l.245El l.245E+l

When entering a decimal zero for either an integer or floating point quantity, the zero can be written in either form. Thus a floating point zero can be entered simply as O without a decimal point. A field starting with a non-blank character other than a digit, sign, comma, period or decimal point, asterisk, dollar sign, slash, or apostrophe is considered a default alphanumeric field. The field is terminated by a comma or the end of the card; all characters except commas are allowed and imbedded blanks are considered part of the alphanumeric field and do not terminate the field. Blanks extending from the last non-blank character of an alphanumeric field to the end of the card are not considered part of the field. (Blanks extending from the last non-blank character are initially treated as fields by the CVI routine, but are later dropped by the INP routine.) An alphanumeric field can also be specified by enclosing the field within apostrophes (’). A blank or comma must follow the terminating apostrophe. The apostrophe field can be used to specify an alphanumeric field beginning with one of the special characters, e.g., ‘6% ENRICHED FUEL’. (Hex and octal fields permitted by CVI are treated as alphanumeric fields by INP.)

Data on a card may be continued on a continuation card by entering a plus sign as the first non-blank character on the continuation card. A field starting on a card must be completed on that card and may not continue to the next card. The plus sign indicating the continuation card is not considered part of the first data field on the continuation card and may be placed alone or adjacent to the first data field. Continuation cards themselves may be continued. In subsequent processing, data on continuation cards are treated as if the data were all entered on one card.

Comment information may follow the data fields on any data card (including cards that are continued) by preceding the comments with an asterisk or dollar sign. A default alphanumeric field preceding a comment must be terminated by a comma or the comment information is considered part of the alphanumeric field.

When card format errors are detected, lines containing a ‘$’ located under the character causing the error and a comment giving the card column of the error are printed. A field containing an error is converted as an alphanumeric field of $$$$$$$$. An error flag is set and input processing continues, but the job can be aborted at the end of input processing. Usually another error is produced by a routine attempting to process the erroneous data.

The first field on a data card is treated as a card number which must be a positive decimal integer number. If the first field has an error or is not a positive decimal integer, the card number is replaced by the current card sequence number, an error statement is printed, and the error flag is set. Data on the card is not used and the card will be identified by the card sequence number if the list of unused data cards is printed. Continuation cards do not have card numbers since they are considered an extension of the first card. After

each card number and the accompanying data are converted, the card number is compared to previously entered card numbers. If a matching card number is found, the data entered on the previous card is replaced by the data of the current card. If the card being processed contains only a card number, the card number and the data entered on the previous card are deleted. If a card causes replacement or deletion of data, a statement is printed indicating that the card is a replacement card.

The list of card numbers and associated data used in a case can be passed to the next case. Cards entered for the next case are added to the passed list or act as replacement cards depending on the card number. The resulting input to cases following the first case is the same as if all previous slash cards were removed from the input to the problem set.

C.2 Programming Use Of The Inp Package

The INP package contains eleven subroutines or entry points which can be called by the programmer using the package. One call is issued to INP for each case in order to read and convert the data for the case, to replace or remove duplicate cards, and to form a sorted table of card numbers cross-referenced to a list. The list contains data words obtained from the cards and mode words generated during input conversion. Because of the sorted table, the order of cards in the data deck is not important. The subroutine INPLNK accesses the table and can locate one card at a time. The subroutine INPMOD is used to check the appropriate mode of the data against a specified list, also one card at a time. The INP2 subroutine is used to check data and to move data from the list to a specified array by using calls to INPLNK and INPMOD. It can be used to process data from a single card or from a set of cards numbered within a specified range, and to check for minimum and maximum numbers of items and appropriate mode. The subroutine INP4 executes repeated calls to INP2 and modifies the call parameters to INP2 by specified amounts. The subroutine INP5 is similar to INP4 in that it can process more than one set of cards but it is more involved in that it expands the input in either sequential or overlay mode and stores the information in either of two orders. Function INP8 can be used to determine whether there are cards in the list that have not been referenced by INPLNK and thus also INP2, INP4, and INP5. The function INP9 deletes cards which have been referenced by INPLNK from the table and list. The function INP10 deletes selected cards from the table and list. Subroutines INP6 and INP7 are used for error comments.

The following sections describe programming requirements for using the INP package. In the following descriptions, calling parameters are named for their integer or floating point format; that is, integer quantities have I, J, K, L, M, or N as their first character and names beginning with any other character are floating point quantities. Integer variables are four byte (32 bit) quantities and floating point variables are eight byte (64 bit) quantities. Symbols appearing in the calling sequences are unique and when the same symbol appears in two or more calling sequences, the symbol has the same definition in each appearance. The symbol is usually completely described only in its first appearance, but the definitions are summarized in the Array and Variable Summaries (Section C.3.2 and Section C.3.3). Calling parameters that are marked with an asterisk both convey information to the subroutine and return information from the subroutine and thus the parameter can have a different value on exit than it had on entry. The error messages referenced in the descriptions are listed in the Error Message Summary (Section C.3.4). Programming errors, such as improper calling parameters, cause an abnormal termination through the FABEND routine.

Input data cards can have a mixture of integer, floating point, and alphanumeric data and the INP subroutine converts and stores all data into the list as eight byte quantities. The INP2, INP4, and INP5 subroutines, although checking the data against a specified list for the correct mode, still moves the data to a specified array as eight byte quantities. The accessing of data from an array containing mixed integer and floating point data is easily accomplished by equivalencing integer and floating point names to the array containing the mixed data, but the storing of integer data as eight byte quantities poses a problem since

integer arithmetic on most 32-bit machines is limited to four bytes. This problem is bypassed by specifying an integer variable as a double subscripted array with the first subscript dimensioned to 2. For example when specifying

```
REAL*8 A(n)
INTEGER IA(2,n)
EQUIVALENCE (A(1),IA(1,1))
```

the i'th quantity can be obtained from A(i) if it is floating point and from IA(2,i) if it is integer.

C.2.1 Input Echo and Package Initialization (INP)

CALL INP (XLl, NLl, NCASE*, NDATA*, ISW)

One call to INP reads all the input cards for the next case. That is, each call to INP reads cards from Fortran Unit 5 from its current position until either a slash card or a period card is read, or the end of the data set is encountered.

The quantity NCASE is incremented by one. NCASE should be set to zero before the first call to INP. If a period card terminates the case, the sign of NCASE is set to minus. The calling program can test the sign of NCASE to determine whether this case is the last case of a problem set or another case follows. If NCASE is negative indicating the end of a problem set, NCASE should be reset to zero before calling INP for the first case of the next problem set.

As input cards are read, they are printed without modification. Before printing the first card, the first heading line is printed at the top of a page and followed by "LISTING OF INPUT DATA FOR CASE n", where n is NCASE after it has been incremented by one. Subsequent pages of the input data listing have only the first heading line at the top of an output page. As title cards are read, the title information with the equal sign removed overlay a title storage array which was initialized to blanks. Just before a normal return, HEADER is called to set the current contents of the title storage array and the date as the second header line, and the two lines of header information are printed at the top of the next output page. Unless other calls are made to HEADER, these two header lines are printed at the top of each succeeding output page.

The parameter XLl is a REAL*8 array of length NLl and is used to store the list and table plus a control word. The control word is stored in XLl(l); the list is stored starting at XLl(2) and extends upward in the array; the table is stored starting at XLl(NLl) and extends downward in the array. The space between the list and table is used for temporary working space. Data card information is converted to binary form through calls to the DCVIC entry to the CVI subroutine (Appendix A). The binary information from a data card that is not a continuation card is stored starting at the beginning of the temporary work space and the mode indicators are stored beginning at the middle of the temporary work space. The binary information and mode indicators for continuation cards are stored following the information for the preceding card. As each individual data card is processed, there must be 40 words between the end of the list or the last converted binary quantity and the beginning of the mode indicators. This space is necessary to prevent the binary quantities from overstoring the mode indicators and the mode indicators from overstoring the table; 40 words are necessary since that is the largest number of quantities that can be entered on an 80 column card. After the data card and any continuation cards have been converted the binary data and mode indicators are stored as if the data were entered on one card and subsequent processing can proceed as if only one card was entered. A table word is then constructed, consisting of the card number or the card sequence number if the card number is illegal, an indicator specifying whether a card number or card sequence number is stored, an indicator specifying whether a card format error was detected, a pointer where the binary information is stored, the number of words on the card other than the card number, and a use indicator which is set off. If there are data other than the card number on the card, the binary

information is moved downward one word eliminating the card number from the list, and the corresponding mode indicators returned from CVI are converted to two bit indicators and stored in a packed form, 30 indicators per word, following the binary information. The table words, one for each card, constitute the table, and the list is made up of the converted binary information and the mode indicators.

The new card number is then compared against the current card numbers stored in the table. If a duplication is found, the table word containing the card number is replaced by the new table word. Space occupied by the replaced list data is retrieved by shifting list data downward over the replaced data when the number of words on the replaced card and the new card are different or simply by overstoring the replaced list data with the new list data when the number of words are the same. Table pointers are updated when list data are moved. When a replacement card contains only a card number, the card acts as a deletion card; the table word is deleted and the list space is retrieved by shifting the list over the deleted data. A message, “CARD ABOVE IS REPLACEMENT CARD”, is printed below any card that replaces or deletes a data card. A card containing only a card number that is not a duplicate card number has no effect on the table and list, and no message is printed.

A normal return from the INP subroutine is made if a period or slash card is read or an end of data set is encountered after at least one input card was read before a normal return, the table is sorted by card number and is moved adjacent to the list and the number of words in the list and the number of words in the table are packed into the control word at XLl(l). Upon normal exit, the absolute value of NDATA is set equal to the number of words needed in XLl to hold the control word, list, and table; the number of words needed is one plus the number of words in the list plus the number of words in the table. The sign of NDATA is set minus if no data cards (cards other than title, comments, slash or period cards) are entered for a case, and in normal usage this indicates that no input data were entered and that a succeeding case may have input identical to the preceding case.

On entry to INP, NDATA indicates whether the array XLl contains data from a previous case. If NDATA is equal to or less than zero, XLI contains nc data from a previous case and the table and list are assumed empty. If NDATA is greater than zero, XLl is assumed to contain data from a previous problem in the same format as that upon exit from INP. That is, XLl contains a control word containing the number of words in the list and table followed by the list and table. The table is moved to the end of XLl with the use indicators in the table set off and the input cards for the current case are then processed as described above.

The parameter ISW indicates the return status. If ISW is zero, a normal return was made and no errors were detected during the processing of the input cards. If ISW is one, the end of data set was encountered when trying to read the first data card of a case and INP returned immediately. If ISW is two, a normal return was made, but card format errors were detected and the list of input cards contains one or more of Error Messages 3 through 6. The usual practice in this case is to continue checking the input data for additional errors but execution is terminated after input checking is completed. If ISW is three, the array XLl is not large enough to process the input data as indicated by Error Message 1 or 2, and INP returned immediately.

C.2.2 Read Dataset (INP2)

CALL INP2 (XLl, XL2, L3)

The array XLl contains the list and table data. The array XL2 is the array into which the data specified in the call is to be moved. The array L3 contains specifications as follows:

L3(1) ICl, first card

L3(2) +IC2, last card number. ICl and IC2 specify the set of card numbers of the data that is to be moved into XL2. If IC2 is zero, only the card with card number ICl is specified.

If IC2 is nonzero, cards with card number c, ICl ≤ c ≤ |IC2| are specified. Not all the cards in the range of c need be present. If IC2 is positive, the card numbers that are present within the range must be sequential and are processed in sequential order. If card numbers ICl, ICl+l, ICl+2, ..., ICl+a are present where ICl+a is the last sequential card number, a card with number cx, ICl+a + 2 ≤ CX ≤ IC2 is an error and causes Error Message 8 to be printed. If IC2 is negative, the cards need not be sequential and are processed in increasing order.

L3(3) MIN, the minimum number of items to be processed. Error Message 9 is printed if fewer items are processed.

L3(4) MAX, the maximum number of items to be processed; ignored if zero. Error Message lO is printed if more items are processed.

L3(5) NJ, the number of words to skip between items in XL2; usually zero.

L3(6) ±J*. J, if positive, is the starting location in XL2; input item n is stored in XL2(J+(N-J+l)*(n-l)). If J is negative, no data is moved, but all checking is performed.

L3(7), ... An array defining the integer, floating point, or alphanumeric format expected on the cards. Format information is defined in INPMOD description (Section C.2.6). Errors cause Error Messages 7, 11, or 12 to be printed.

INPLNK is used to locate each card and INPMOD is used to check the format.

On exit, L3(6) or J is set to NMOVE, the number of items moved, if no errors were found and J was positive on entry. L3(6) or J can be zero indicating no cards with the specified card numbers are present if L3(3) or MIN is set to zero. If any card requested contained a card format error (as detected by INP) or if any of the tests specified in the L3 array failed, L3(6) or J is set to -l on exit. If J was negative on entry, L3(6) or J is set to -NMOVE on exit if no errors were found. Only the first error in the specified set of cards is found and processing is terminated after the appropriate error message is printed. Error Message 13 and 14 can be printed by INP2 in addition to those noted in the definition of L3.

C.2.3 Multiple Dataset Reads (INP4)

CALL INP4 (ICl, ±IC2, MIN, MAX, NJ, J*, IC3, NTIMES, NEWJ, XLl, XL2, L5)

Subroutine INP4 makes NTIMES calls on INP2. An abnormal termination occurs if NTIMES is zero or negative. For the first call on INP2, ICl, IC2, MIN, MAX, NJ, and J are as described for INP2 and L5 is the format array for checking the mode of the data. For the following calls, ICl and IIC21 are increased by IC3, and J if positive is increased by NEWJ. J is changed upon exit as in the INP2 description. ICl and IC2 on exit have the same value as on entry. Maximum size of the LS array is 40.

C.2.4 Read Vector Dataset (INP5)

CALL INP5 (ICl, ±IC2, IC3, ±Nl, ±NMIN,±NMAX, ±NSTORE, NTIMES, NEWJ, J*,XLl, XL2, L5, XL6, NL6)

The subroutine INP5 is similar to INP4 in that it makes NTIMES calls on INP2, but is more powerful in that it accepts self-expanding data of the sequential or overlay type. The basic unit of input data is a vector, S, of |N1| components. If Nl is positive, the data is of sequential form where (S_k,n_k), k=l...K means that S_k is to be repeated n_k - n_{k-1} times and represents expanded vectors n_{k-1} + 1 through n_k with n_O = |NMIN|, n_k > n_{k-1}, and n_k ≤|NMAX|. If Nl is negative, the data is of the overlay form where (m_k,S_k,n_k), k=l,... K where S_k is overlaid on an initial set of vectors beginning at the m_k'th vector and extending

through the n_k'th vector, with $m_k \leq n_k$, min $m_k \geq$ |NMIN|, and max $n_k \leq$ |NMAX|. For either type there results a sequence of expanded input of the form S_i, $i_O \leq i \leq$ |NMAX|, where certain of the S_i may be missing in case of overlay expansion. Here i_O = |NMIN| + l or |NMIN| for sequential and overlay types respectively. For overlay data, a positive **NMIN** requires that the lower limit be included while a negative NMIN only specifies a lower bound. For sequential type, a negative NMIN can be used for negative indexing. For both types a positive NMAX is used to require that the upper limit be included while a negative NMAX only specifies an upper bound.

The vectors, S_i, form a two dimensional array with elements, $s_{k,i}$, where $l \leq k \leq$ **|N1|, |NMIN|** $\leq$ **i** $\leq$ **|NMAX|.** This array can be stored into XL2 in two different modes where one mode is the transpose of the other mode. If NSTORE is positive, $s_{k,i}$ is stored in XL2(J+(k-1) + NSTORE*(i-l)) and |NSTORE| should be greater than or equal to |N1|. If NSTORE is negative, $s_{k,i}$ is stored in XL2(J+(i-l) + |NSTORE|*(k-l)) and proper size of NSTORE depends on NMIN and NMAX. This is equivalent to having a two dimensional array defined as REAL*8 SS(NSTORE,n) and EQUIVALENCE (XL2(J),SS(l,l)) and if NSTORE is positive, $s_{k,i}$ is stored in SS(K,I), and if NSTORE is negative, $s_{k,i}$ is stored in SS(I,K).

ICl, IC2, IC3, NTIMES, J, NEWJ, and L5 are handled as in INP4. J on exit contains not the amount of expanded data, but the amount of data on the cards. The array XL6 of length NL6 is used for temporary storage and must be large enough to hold the unexpanded input data for one card set ICl$\leq$ c $\leq$ IC2. Additional error checks are made because of the form of the input and Error Messages 15 through 20 can be printed. Subroutine INP6 is called for error processing. If the parameters, Nl, NSTORE, or NTIMES, are less than or equal to zero, an abnormal termination occurs.

C.2.5 Card Number Search (INPLNK)

CALL INPLNK (IC, IX, N3, N4, XLl)

Subroutine INPLNK searches the table and list array, XLl, for card IC. The subroutine exits with IX equal to the card number in the table next larger than IC unless such a card does not exist and then IX equals -l. On exit, if N4 equals 0, card IC is not in the table; if N4 < 0, a format error detected by INP is on card IC; and if N4 > O, there are N4 data fields excluding the card number on card IC and the data are stored sequentially beginning at XLl(N3). A use flag is set on in the table entry for IC if it is found. INPLNK issues no error messages.

C.2.6 Data Type Identification (INPMOD)

CALL INPMOD (XLl, L3, N3, N4, N5, N6)

Subroutine INPMOD checks N4 items of data stored sequentially beginning at XLl(N3) for appropriate format. The format specification begins at L3(7); the specification starting at 7 is consistent with the format specification for INP2. The format entries are -1 for alphanumeric, O for integer, and l for floating point. Cyclic repetition of two or more entries can be condensed by prefixing the repeated format by N, where the magnitude is the number of items repeated, the sign is positive if the cycle is to be reset for each entry, and the sign is negative if it is to pick up at the stopping point of the previous entry.

To allow starting within the format specification, N6 is the number of items previously checked with the current format. On exit, N5=0 if no errors were found; 0 < N5 < l0000 if item N5 should have been integer but was not; N5 < 0 if item -NS should have been floating point, but was not; and N5 > l0000 if item NS-l0000 should have been alphanumeric, but was not. A decimal zero is considered either integer or floating point as required to satisfy mode tests. INPMOD issues no error message. INPMOD calls INPUPK which is part of the INP package.

C.2.7 Error Processing (INP6 and INP7)

CALL INP6 (ICl, IC2, N2, ICARD, ITEM, XLl)

Subroutine INP6 can be entered when the program using the INP package finds that the N2'th item of a set of cards ICl ≤ c ≤ IC2 processed by INP2 was in error. On exit, ICARD is the card number and ITEM is the number of the field on the card containing the error.

CALL INP7 (ICARD, ITEM)

Subroutine INP7 simply prints Error Message 22 stating that item ITEM on card ICARD is in error. This subroutine can be used to print error information obtained from INP6. This subroutine is not called by other INP package routines.

C.2.8 Count Unprocessed Cards (INP8)

FUNCTION INP8 (NPRINT, XLl)

Function INP8 returns the number of cards that have not been processed by INPLNK, INP2, INP4, or INPS. This is done by counting the number of table entries in XLl that do not have the use flag set on. If NPRINT is l, the card numbers of the unprocessed cards are listed under Error Message 21, while if NPRINT is 0, no output is printed. The list of unprocessed cards will include any cards that have an invalid card number; the card sequence number is printed in place of the card number and the sequence number is preceded by asterisks to distinguish it from a card number.

C.2.9 Delete Processed Cards (INP9 and INP10)

FUNCTION INP9 (XLl)

FUNCTION INP10(XLl, ICl, IC2)

Functions INP9 and INP10 delete table entries and associated data and mode information from the array XLl. INP9 deletes cards that have been referenced at least once by INPLNK, INP2, INP4, or INPS, that is, table entries with the use bit set on. INPl0 deletes all cards, IC1 ≤ c ≤ IC2. When cards are deleted, all holes created by deletion are squeezed out, the table entries are adjusted accordingly, and the control word is updated. The functions return the new length of the table, list and control word. When all cards have been deleted, the length required in the XLl array is one--the length required for the control word.

When a card is deleted, a hole in the table and list is created, and the remaining tables and data must be moved to regain the storage made available. In order to move the remaining table and list only once and not use storage outside the array XLl, the holes are marked with the bit pattern 7FFFFFFFFFFFFFF as they are formed. After all cards are deleted, holes are located by testing for the special bit pattern. Thus, the bit pattern used for marking holes must not be allowed as a data item. The bit pattern selected is unlikely to occur. As described for INP (Section C.2.1) if that bit pattern is entered as input data, the last significant bit is set to zero.

C.3 INP Summary

Following are the INP package calls, summary of parameters, list of error messages, and structures of control word, table words, and mode words.

C.3.1 Summary of INP Package Calls

CALL INP (XLl, NLl, NCASE*, NDATA*, ISW)

CALL INP2 (XLl, XL2, L3)

CALL INP4 (IC1, +IC2, MIN, MAX, NJ, J*, IC3, NTIMES, NEWJ, XLl, XL2, L5)

CALL INP5 (IC1, +IC2, IC3, +N1, +NMIN, +NMAX, +NSTORE, NTIMES, NEWJ, J*,XLl, XL2, L5, XL6, NL6)

CALL INPLNK (IC, IX, N3, N4, XLl)

CALL INPMOD (XLl, L3, N3, N4, N5, N6)

CALL INP6 (ICl, IC2, N2, ICARD, ITEM, XLl)

CALL INP7 (ICARD, ITEM)

FUNCTION INP8 (NPRINT, XLl)

FUNCTION INP9 (XL1)

FUNCTION INP10 (XL1, ICl, IC2)

C.3.2 Array Summary

L3	Array used for specifications to INP2 with L3(1) = IC1, L3(2) = +IC2, L3(3) = MIN, L3(4) = MAX, L3(5) = NJ, L3(6) = J*, L3(7) and above equivalent to L5(1) and above.
L5	Array used to define appropriate mode of data fields on card or set of cards excluding the card numbers. An entry of -l is for alphanumeric fields, an entry of O is for integer fields, and an entry of l is for floating point fields. If a format repeats beyond a point, prefix the repeated format in L5 by +N, |N!, 2, where N is the number of items repeated. Use N positive to reset the repeat cycle at the beginning of each card, or use N negative to allow the cycle to overlap cards. Within a cycle, all elements must be O or ±l, and only one cycle is allowed. Size of L5 array is 40. Array for INP2 starting at L3(7) is not limited.
XLl	Array containing control word, converted data from cards, mode indicators, and table entries.
XL2	Array into which data is to be moved.
XL6	Array for temporary storage used in INP5.

C.3.3 Variable Summary

IC	Card number desired.
ICl,+IC2	Define card numbers of a set, ICl < c < |IC2|. If IC2 is zero, only one card, ICl, is requested. If IC2 < O, card numbers must be sequential, c = ICl, ICl+l, ..., ICl+a ≤ IC2, and if ICl+a is the last card in sequence, the next larger card CX must not be in the range ICl+a+2 < c < IC2. If IC2 is negative, cards need not be sequential and are taken in increasing order. As used for INP10, all cards c, ICl < c < IC2 are deleted.
IC3	Added to ICl and IIC2l to define next set for use in INP4 and INP5.
ICARD	Number of card containing error item N2.

IX	Card number in table next larger than IC unless no such card is present, then IX is returned as -l.
INP8	Result of function call is the number of cards in table not processed by INPLNK, INP2, INP4, or INP5.
ITEM	Item number on CARD of item N2 on set of cards in error.
J	On entry, if J is positive, store data beginning at XL2(J); if J is negative, do not move data into XL2 but check data. On exit, J is set to -l if an error was found; if positive on entry and no errors were found, it is set to +MOVE; if negative on entry and no errors were found, it is set to -NMOVE.
MAX	Maximum number of data items in a set of cards.
MIN	Minimum number of data items in a set of cards.
+Nl	Nl = Number of elements in the input vector S; data is sequential type if Nl is positive and overlay type if Nl is negative.
N2	The number of the data field in error in call to INP6.
N3	On return from INPLNK, XLl (N3) contains first data word if card is present.
N4	Number of data words on card IC located by INPLNK if N4 is positive; if N4 is zero, card IC is not in table; if N4 is negative, format error was found on card IC.
N5	On exit from INPMOD, N5 = 0 if format correct; O < NS < l0000 if item N5 should have been integer but was not; N5 < 0 if item N5 should have been floating point, but was not; and N5>l0000 if item N5-10000 should have been alphanumeric but was not.
N6	On entry to INPMOD, the number of previously checked items. This is used to locate proper starting position in L5.
NCASE	On entry, previous case number which should be non-negative and zero if the first case. On exit, is the current case number, negative if the last case of a problem set.
NDATA	On entry, NDATA > 0 calls for adding data to previous list and table in XLl, and NDATA = 0 indicates no previous list and table is present and the new data is complete in itself.
NEWJ	Added to J for subsequent set in INP4 and INP5.
NJ	Number of words to skip in XL2 between items. INP2 stores data item n in XL2(J + (NJ+l)*(n-l)). Usually NJ is zero.
NLl	Size of XLl on entry to INP.
NL6	Size of XL6; must be large enough to hold data from one set in call to INP5.
+NMAX	Upper limit for sequential and overlay data; if NMAX is positive limit must be included.
+NMIN	If sequential type, NMIN = n0; if overlay type, min $m_k \geq$ NMIN, with NMIN positive requiring that the lower limit be included.
NPRINT	If 1, list unprocessed card numbers, if zero, do not list them.

+NSTORE Used to control storage of data in INP5.

NTIMES Number of sets of cards to process in call to INP4 or INP5.

C.3.4 Error Message Summary

All error comments are preceded by eight asterisks to facilitate recognition of error comments in the midst of regular program output. The lower case letters represent call parameters or symbols used in the subroutine descriptions and actual values are substituted in the output.

l. INSUFFICIENT STORAGE ALLOCATION FOR PREVIOUS DATA, PROCESSING TERMINATED.

2. INSUFFICIENT STORAGE FOR DATA, PROCESSING TERMINATED.

3. $ (placed under column in error) $ POINTS TO CARD ERROR AT COL.

4. END OF FILE ENCOUNTERED BEFORE END(.) CARD.

5. CONTINUATION CARD INDICATED, BUT NO PREVIOUS DATA CARD. TREATED AS NEW DATA CARD.

6. UNRECOGNIZABLE CARD NUMBER

7. WORD n5 ON CARD ic SHOULD BE IN ALPHANUMERIC FORMAT

8. CARD c+a+l MISSING IN SEQUENCE

9. TOO FEW NUMBERS ON CARDS icl THROUGH ic2

l0. TOO MANY NUMBERS ON CARDS icl THROUGH ic2

11. WORD n5 ON CARD ic SHOULD BE IN INTEGER FORMAT

12. WORD n5 ON CARD ic SHOULD BE IN FLOATING POINT FORMAT

13. CARDS ic1 THROUGH ic2 MISSING

14. ILLEGAL FORMAT ON CARD ic

15. m NUMBERS ON CARDS icl THROUGH ic2 ARE NOT A MULTIPLE OF nl

16. ITEM m ON CARD ic IS LESS THAN MINIMUM ALLOWED OF nmin

17. ITEM m ON CARD ic EXCEEDS MAXIMUM ALLOWED FOR nmax

18. ERROR IN LIMITS OF THE SET BEGINNING AT ITEM m ON CARD ic

l9. LOWER LIMIT OF nmin NOT INCLUDED ON CARDS icl THROUGH ic2

20. UPPER LIMIT OF nmax NOT INCLUDED ON CARDS icl THROUGH lc2

21. THE FOLLOWING CARDS WERE NOT USED

22. ITEM item ON CARD card IN ERROR

C.3.5 Word Structure Used in XLl

C.3.5.1 Control Word Structure

The control word stored in XLl(l) consists of two 32 bit integer words. Integer word l contains the length of the table; integer word 2 contains the length of the list containing the binary data and the mode indicators.

C.3.5.2 Table Word Structure

Bits are numbered from the left, starting with 0.

Bit Use flag; O if card not processed, l if processed.

Bits 1-29 Card number or sequence number.

Bit 300 if bits 0-29 contain card number, l if sequence number.

Bit 310 if no format errors on card, l if errors.

Bits 32-47 Index in XLl of first data word of card associated with this table entry.

Bits 48-63 Number of words on card excluding card number. Card numbers are limited to 536,870, 911 and card numbers greater than this cause Error Message 6 to be printed. Half word arithmetic is used for the index and word count and this limits the list length to 32,766 words; this limit is not checked and can be removed if necessary.

C.3.5.3 Mode Indicator Word Structure.

For each card (continuation cards here considered as part of the first card), the mode indicator words are stored immediately following the last data word. Mode indicators consist of two bits and these are packed left justified, 30 indicators per word. Because packing only 30 indicators per word leaves the last four bits zero, there is no possibility for a mode indicator word to be treated as the special bit pattern used for deleting card data. The mode indicators are O for an integer or floating point zero, l for a nonzero integer, 2 for a nonzero floating point quantity, and 3 for an alphanumeric quantity.

Appendix D. THERMODYNAMIC PROPERTY PACKAGE

D.1 Introduction

STH20 is a code package that can be used to compute thermodynamic properties of water. The package consists of a module that generates tables of water properties and writes the tables on a data set, a subroutine that retrieves the tables, and several subroutines that compute the thermodynamic properties by table search and interpolation procedures. The module need be executed only once if the tables are stored in a permanent data set. The subroutines can be readily incorporated into programs requiring water properties. The subroutine that retrieves the table is usually executed only once each time a module incorporating the subroutines is invoked unless the storage space for the table is overstored. The remaining subroutines that compute the water properties differ in the quantities specified as input. Water properties are calculated given the input values of temperature and quality, pressure and quality, temperature and pressure, temperature and specific volume, or pressure and enthalpy. The properties returned by the subroutines include temperature, pressure, specific volume, internal energy, enthalpy, thermal expansion, compressibility, and heat capacity.

This package is considered only an interim solution to the problem of computing water properties but, hopefully, the routines can be used until improved water property routines can be developed. The main disadvantage of the package is the large amount of core storage required for the tables; 7628 64-bit words are required for the tables listed at the back of this report. The accuracy is dependent on the table density and the adequacy of the interpolation procedures. The accuracy can be increased, but at the penalty of increased core storage requirements. The interpolation procedures are such that no discontinuities should be introduced into the properties and the derivative quantities should be smooth since these quantities are included in the tables rather than obtained from table differencing.

The subroutines use SI units for all quantities. Listed below are the thermodynamic quantities used in the package, the symbols used for the quantities, and their units. It should be noted that each quantity may have a subscript f, denoting saturated fluid, or subscript g, denoting saturated gas.

Table D-1: STH20 Thermodynamic Quantities

Quantity	Symbol	Units
Temperature	T	Kelvin (K)
Pressure	P	Pascal (Pa) = Newton/meter^2 (N/m^2)= Joule/meter^3 (J/m^3)
Specific Volume	v	meters^3/kilogram (m^3/kg)
Internal Energy	u	Joule/kilogram (J/kg)
Enthalpy	h	Joule/kilogram (J/kg)
Thermal Expansion	β	Kelvin $^{-1}$ (K $^{-1}$)
Compressibility	κ	Pascal^{-1} (pa^{-1})
Heat Capacity	c_p	Joule/kilogram-Kelvin (J/kg-K)

D.2 STH20G, A Module to Generate Tables of Water Properties

The STH20G module generates tables of thermodynamic properties of water. Five tables are generated and packed into one array. The first two tables are temperatures and pressures obtained from input data; the third table contains saturation properties as a function of the saturation temperatures in table one; the fourth table is a separate saturation table as a function of the saturation pressures in table two; and the fifth table is a two-dimensional table containing the single phase properties as a function of the temperatures and pressures in tables one and two. The tables are written on a data set for use by other modules utilizing the subroutines. The tables are generated and stored using 64-bit words for floating point quantities.

D.2.1 Input Data

The input data is in the INP format (see Appendix D) and only one case is allowed.

D.2.1.1 Temperature and Pressure Count Card

Card No.	Data
1000	NT, NP where NT is the number of temperatures entered and NP is the number of pressures entered. Both quantities are in integer format.

D.2.1.2 Temperature Cards

Card N	Data
1001-1999	Temperature values are entered in floating point format in increasing order on cards with increasing card numbers, one or more temperatures per card. Temperatures must be in the range $273.16\ K \leq T \leq 1073.15\ K$. The card numbers need not be sequential and this facilitates the addition or deletion of temperatures. For proper operation of the water property subroutines, the temperatures 273.16 K (triple point), 647.30 K (critical point), and 1073.15 K (upper limit of ASME formulation) should be entered. A warning message is issued if any of these temperatures is missing, but execution continues.

D.2.1.3 Pressure Cards

Card No.	Data
2001-2999	Pressure values are entered in floating point format in increasing order on cards with increasing card numbers, one or more pressures per card. Pressures must be in the range $O < P < 108$ Pa. The card numbers need not be sequential. For proper operation of the water property subroutines, at least one pressure should be above $2.212 \cdot 10^7$ Pa (critical pressure). A warning message is issued if the last pressure is not above the critical pressure.

D.2.2 Table Format

The five tables are packed into a single dimensioned array. In the description of the tables, A is used for the array symbol, NT and NP are defined in the input description, NS is the number of input temperatures not above the critical temperature, and NS2 is the number of input pressures not above critical pressure.

D.2.2.1 Temperature Table

The temperatures in increasing order as obtained from the input data are stored in A(l) through A(NT). The temperatures can be considered to be stored in an array dimensioned T(NT), where T(l) is equivalenced to A(l).

D.2.2.2 Pressure Table

The pressures in increasing order as obtained from the input data are stored in A(NT + 1) through A(NT + NP). The pressures can be considered to be stored in an array dimensioned P(NP), where P(l) is equivalenced to A(NT + 1).

D.2.2.3 Saturation Table as a Function of Temperature

The saturation properties as a function of temperature are stored in A(NT + NP + 1) through A(NT + NP + NS*11). The saturation properties are stored as an array dimensioned B(ll,NS), where B(l,l) is equivalenced to A(NT + NP + 1). The saturation values in B(N,I), 1 < N < 11, are a function of the temperature in T(I). The correspondence between the B array and the saturation properties are shown in the following table.

Table D-2: Steam Table Interface

B(l,I)	P
B(2,I)-	V_f
8(3,I)	U_f
B(4,I)	β_f
B(5,I)	κf
B(6,I)	C_{pf}
B(7,I)	v_g
B(8,I)	u_g
B(9,I)	β_g
B(l0,I)	κ_g
B(11,I)	c_{pg}

D.2.2.4 Saturation Table as a Function of Pressure

The saturation properties as a function of pressure are stored in A(NT + NP + NS*11 + 1) through A(NT + NP + NS*11 + NS2*11). The saturation properties are stored as an array dimensioned C(11,NS2), where C(l,l) is equivalenced to A(NT + NP + NS*11 + 1). The saturation values in C(N,J), 1 < N < 11, are a function of the pressure in P(J). The correspondence between the C array and the saturation properties is the same as for the B array except that C(l,J) is the saturation temperature instead of the saturation pressure.

D.2.2.5 Single Phase Property Table

The single phase properties as a function of temperature and pressure are stored in A(NT + NP + NS*11 + NS2*11 + 1) through A(NT + NP + NS*11 + NS2*11 + NT*NP*5). The single phase properties are stored as an array dimensioned D(5,NT,NP), where D(l,l,l) is equivalenced to A(NT + NP + NS*11 + NS2*11 + 1). The values D(N,I,J), 1 < N< 5, are a function of the temperature in T(I) and the pressure in P(J). The correspondence between the D values and the properties are:

Table D-3: Single Phase Property Interface

D(l,I,J)	ν
D(3,I,J)	υ
D(2,I,J)	β
D(4,I,J)	κ
D(5,I,J)	c_p

D.2.3 Module Output

The tables generated by the module are written on disk in two records using Fortran sequential, non-formatted writes. The first record contains NT, NP, NS, and NS2. The second record contains the array A containing NTOT elements where NTOT = NT + NP + NS*11 + NS2*11 + NT*NP*5. The printed output includes the input data, values of NT, NP, NS, NS2, and NTOT, and the generated table values. Output from an execution of the module is attached to the back of this description.

D.3 STH2OI, Initialization Subroutine

The STH2O1 subroutine retrieves the tables needed by the other subroutines by reading the data set created by the STH20G module and sets up a small common block named STH20C. This subroutine must be called before a call to **any** of the other subroutines. The subroutine is called by CALL STH20I (A, N, NUSE) where A is a 64 bit floating point array of length NUSE available for storing the tables and N is the unit number to be used to read the data set. On entry, NUSE must be greater than or equal to the length of the generated table (NTOT). On return, NUSE is set positive and contains the length of the tables if the tables were successfully loaded into the array A. The value returned in NUSE is thus the number of 64 bit words used in A and words in A beyond A(NUSE) are available for other use. The subroutine prints an error message and return NUSE equal to -1 if the table cannot be retrieved. The subroutine can be placed in an overlay. The subroutine need be executed only once for each invoking of a module containing the STH20 subroutines unless the array A is destroyed.

D.4 Arguments Common To The Remaining Subroutines

The array A is defined in the description of the STH20G module and must be the array loaded by the call to STH20I. The integer variable IT is set by the subroutines that can compute properties in the liquid, two phase, and vapor states. IT can be set to 1, 2, 3, or 4 as shown in Figure E-1 and is used to indicate whether a saturated pressure (or temperature) corresponding to an input temperature (or pressure) is available. Figure E-1 also indicates the value of quality returned for the various states and whether the fluid is considered liquid or vapor for interpolation procedures. Water above the critical pressure but below the critical temperature is considered in the liquid state.

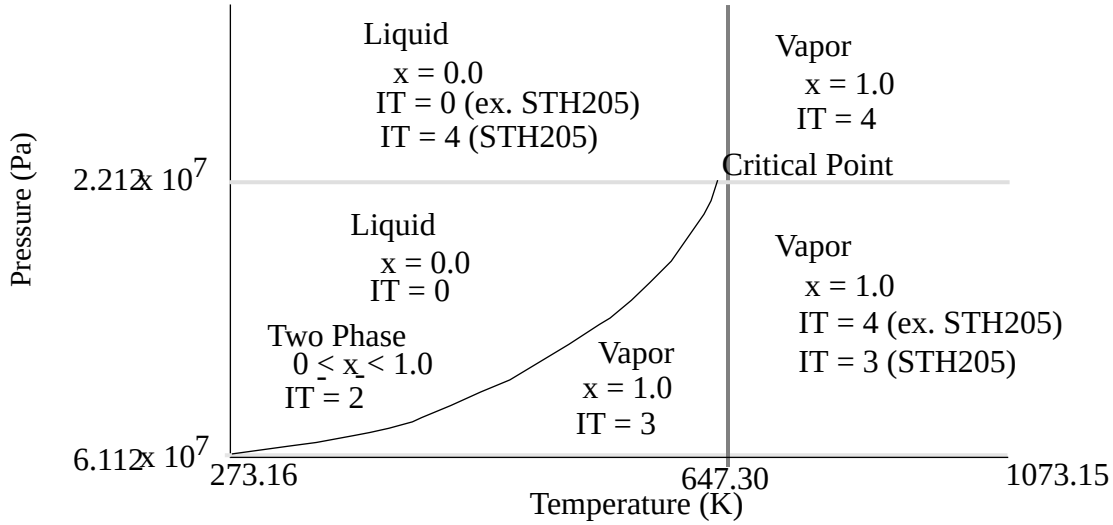

Figure D-1. Sketch Showing Definitions of State, Quality, and IT.

S is an array of 23 64 bit floating point words containing both input to and output from the subroutines. The assignment of properties to the S array is :

Table D-4: Arguments to Steam Table Routines

S(l)	T
S(2)	P
S(3)	ν
S(4)	μ
S(5)	h
S(6)	β
S(7)	κ
S(8)	c_p
S(9)	x (quality)
S(10)	P_{sat} or T_{sat}
S(ll)	v_f
S(l2)	v_g
S(13)	u_f
S(14)	u_g

Table D-4: Arguments to Steam Table Routines (Continued)

S(15)	h_f
S(16)	h_g
S(17)	β_f
S(18)	β_g
S(l9)	κ_f
S(20)	κ_g
S(21)	c_{pf}
S(22)	c_{pg}
S(23)	indexes

S(6) through S(8) are undefined for the two phase state (IT is equal to 2). S(9) contains 0.0 for the liquid state (IT may be 1 or 4), contains the quality if in the two phase state (IT is equal to 2), and contains 1.0 for the vapor state (IT may be 3 or 4). For all subroutines except STH2O5, S(10) is the saturated pressure corresponding to the temperature in S(l) if IT is returned 1 through 3 and is undefined if IT is returned as 4. For STH205, S(10) is the saturation temperature corresponding to the pressure in S(2) if IT is returned as 1 through 3 and is undefined if IT is returned as 4. S(11) through S(22) are undefined for single phase states (IT is not equal to 2). The S array is used for working storage and undefined elements may be changed during subroutine execution. On entry, the indexes in S(23) are used to start the table search if they are valid, and on return contain the indexes obtained by the table search. Execution time can be saved if indexes close to the final values can be supplied as input. This can be accomplished in iteration and time advancement procedures by appropriate saving and restoring of the indexes. The subroutines do not fail when invalid data is entered in S(23) since the table search starts at the beginning of the table in that case.

ERR is returned FALSE if the input quantities are within range of the tables and TRUE otherwise.

D.5 Saturation Pressure as a Function of Temperature

CALL STH200 (T, P, ERR) computes the saturation pressure P given the temperature T as input. The temperature must be in the range 273.16K < T < 647.30K. This subroutine does not use the array A and can be called before STH20I is called. STH200 is an entry point in the STH201 subroutine.

D.6 Saturated Properties as a Function of Temperature and Quality

CALL STH201 (A, S, ERR) computes the saturated water properties given temperature and quality as input. The temperature is entered in S(l) and must be greater than or equal to either T(l) or C(l,l) and less than or equal to either T(NS) or C(l,NS2). The arrays T and C are defined in D.2. These values would be 273.16K and 647.30K, respectively, if the tables were generated without any warning messages. The quality is entered in S(9) and must be in the range 0.0 < x < 1.0. S(3) through S(5) return values for the two phase mixture and S(11) through S(22) return values for saturated liquid and saturated vapor. S(2) and S(10), the saturated pressure corresponding to S(l), are returned equal. IT would always be 2 for this call and thus is not included in the argument list.

D.7 Saturated Properties as a Function of Pressure and Quality

CALL STH202 (A, S, ERR) computes saturated water properties given pressure and quality as input. The pressure is entered in S(2) and must be in the pressure range of saturated water 611.2444 Pa < P < $2.212 \cdot 10^7$ Pa, and have a corresponding saturation temperature limited as described in D.6 above. The saturation temperature is returned in S(l) and the other elements of S are set as in STH20l. STH202 is an entry point in the STH20l subroutine.

D.8 Single Phase Properties as a Function of Temperature and Pressure

CALL STH203 (A, S, IT, FRR) computes single phase water properties given temperature and pressure as input. The temperature is entered in S(l) and must be within the range T(l) < S(l) < 5000 K. The pressures entered in S(2) and must be within the range O < S(2) < P(NP) for the vapor state and P(l) < P < P(NP) for the liquid state. The arrays T and P are defined in D.2. IT is never set to 2 since temperature and pressure cannot determine a two phase condition. The single phase quantities are returned in S(3) through S(8) and S(9) is set to either 0.0 or 1.0 corresponding to the liquid or vapor state. S(10) is the saturation pressure corresponding to S(l) if IT ≠ 4. See also Figure E-1

D.9 Water Properties as a Function of Temperature and Specific Volume

CALL STH204 (A, S, IT, ERR) computes water properties given temperature and specific volume as input. The temperature is entered in S(l) and must be within the range T(l) < S(l) < 5000 K. The range of specific volume depends on the state. If the specific volume indicates the liquid state, the resultant pressure must be within the range P(l) < S(2) ≤ P(NP) and if the specific volume indicates the superheated state, the resultant pressure must be less than P(NP). The two phase region is within range if the temperature is within range. The T and P arrays are defined in D.2. The results are stored in S as described in D.4 and D.8.

D.10 Water Properties as a Function of Pressure and Enthalpy

CALL STH205(A, S, IT, ERR) computes water properties given pressure and enthalpy as input. The pressure is entered in S(2) and the enthalpy is entered in S(5). The range of pressure is identical to that for STH203 and the enthalpy must be in a range that would be returned by valid call to STH203. The results are stored in S as described in D.4. S(10) is the saturation temperature corresponding to S(2) if IT ≠ 4. See also Fig. E-1.

D.11 Interpolation Procedures

The STH20l, STH202, STH203, STH204, and STH205 subroutines all use similar table search and interpolation procedures. Weighted derivative interpolation is used for liquid specific volume and linear interpolation is used for the other liquid properties. Linear or reciprocal interpolation is used for vapor properties based on the perfect gas relationships, Pv = RT, u = C_VT, h = C_pT, $\beta = T^{-1}$, and $\kappa = p^{-1}$. For example, since v = RTP^{-1}, linear interpolation is used for the temperature dependence and reciproca1 interpolation is used for the pressure dependence. The linear interpolation is

$$Y = Y_1 + \frac{(Z - Z_1)\ (Y_2 - Y_1)}{(Z_2 - Z_1)}$$

the reciprocal interpolation is

$$Y = Y_1 + \frac{(Z - Z_1)\ (Y_2 - Y_1)}{(Z_2 - Z_1)\, Z}$$

.

Appendix E. RSTPLT AND STRIPF FILE FORMATS

All versions of SCDAP/RELAP5 follow the same philosophy for the file formats. That philosophy allowed the number of files and their length to be problem dependent. As new versions of the code were developed, additional files were added, but the method of controlling their writing and reading remained the same.

The files are written/read using either buffer out/buffer in or write/read statements. The define variable, bufr, defines which are used. If bufr is defined, the buffer out/buffer in statements are used. This option is normally used only on CRAY-1, CRAY-XMP, CRAY-YMP, but not the CRAY-2. The buffer out/buffer in are limited to writing/reading contiguous blocks of memory, but they have two useful features. One is that the buffer out/buffer in statements only initiate an input/output operation and then return to execute the following statements. Thus the input/output operation can execute in parallel with other operations. The input/output operation is forced to completion or synchronized by a call to the unit function. The second feature is that on input, it is possible to attempt to read more information that exists in a record, and to use the length function to find the amount of data read. If bufr is undefined, unformatted write/read statements are used. The data are written from or read into contiguous locations in the same manner as if the buffer out/buffer in statements were being used. To provide capability similar to the length function available with the buffer out/buffer in statements, two write/read operations are used in place of each buffer out/buffer in operation. The first write statement writes a two integer (eight byte total) record. The first word is the number of words in the record that follows. The second word is four or eight to indicate the number of bytes per word in the following record. All plot related records use eight byte words. All RELAP5 related restart records use eight byte words, but the SCDAP restart records use four byte words if the four byte define option is used. During read operations, the two word record is read to obtain the record length and word size to control the following read of the data. Data transferred to the RSTPLT file and unformatted STRIPF file are integer or real data. Character data are converted to real (or real *8) format by incore write statements before writing to the files.

E.1 RSTPLT and STRIPF file handling

E.1.1 File Initialization (RRSTD)

This subroutine opens the RSTPLT file and writes the first record for new problems.

E.1.2 File Positioning (RESTF)

This subroutine opens the RSTPLT file in restart problems and reads the file until the proper restart records have been read. If the restart problem switches from steady state to transient or vice versa, the restart file is rewound and the first record is rewritten.

E.1.3 Write Plot Data (WRPLID)

This subroutine writes the plotinf, plotalf, and plotnum records. These records are always the second, third, and fourth records of a RSTPLT file, and may be written again at a restart if changes in the plot records occur due to changes made at restart.

E.1.4 Restart Write (RSTREC)

This subroutine writes the restart records.

E.1.5 Plot Write (PLTWRT)

This subroutine gathers the plot data into a buffer for writing. On 32 bit machines, the plot data is compacted from 64 bit floating point (real*8) to 32 bit (real*4) format in this subroutine.

E.1.6 Data Compaction (DTSTEP)

This subroutine compacts the data from 64 bit to 32 bit format for 64 bit machines, and writes the plot records for all machines. This subroutine performs many other functions related to time step control and control of output.

E.1.7 File Close (TRNFIN)

This subroutine closes the RSTPLT file.

E.1.8 Strip File Open (SRESTF)

This subroutine is used for strip problems and opens the RSTPLT and STRIPF files, reads the first record of the RSTPLT file, and writes the first record of the STRIPF file.

E.1.9 Strip File Generation (STRIP)

This subroutine reads the remainder of the RSTPLT file, uses the plotinf, plotalf, and pltnum files to locate the user specified data in the plot records, and writes the appropriate plotinf, plotalf, plotnum, and plotrec records on STRIPF.

E.2 Restart Structure

The RSTPLT file is used for both input and output. A cautious user might backup the RSTPLT file before attempting a restart. If the restart simply continues from where the previous problem terminated, there is little danger of somehow ruining the data in the RSTPLT file. If the restart problem restarts at other than the end of the RSTPLT file, data from the point of restart is overwritten. The restart file must be copied if both problems are to be either restarted or plots obtained.

The RSTPLT and STRIPF files use the same philosophy. But the STRIPF is simpler because no restart records are written to the STRIPF file and the plotinf, plotalf, and plotnum records are written only once to the STRIPF file.

The first record of both files contain ten eight byte words. The first three words contain code identification information; the next three words contain either ‘restart plot file’ or ‘strip file’; the next three words contain date and time information; the last word contains a flag indicating steady state or transient mode.

The plotinf record contains three words. The first word is ‘plotinf’. The second word contains the length of the plotalf, plotnum, and the plotrec (after expansion) records. The third word is zero if the plot records are not compressed, and the length of the plot records if they are compressed. All versions later than MOD2 use compressed plot records in the RSTPLT file. Plot records are not compressed in the strip file.

The plotalf record contains the number of words given in plotinf. Word 1 contains ‘plotalf’. The remaining words contain the alphanumeric part of the two word name of a plot quantity.

The plotnum record contains the same number of words as plotalf. Word1 contains ‘plotnum’. The remaining words contain the numeric part of the two word name of a plot quantity.

The information in plotalf and plotnum records are such that the alphanumeric quantity in the n'th word of plotalf together with the numeric quantity in the n'th word of plotnum form the two word name of the quantity appearing in the n'th word of each plotrec record (after expansion). The names of the plot quantities are given in the users manual. The plotinf, plotalf, and plotnum records are always written as consecutive records. This set of three records always follow the first record of both RSTPLT and STRIPF files. This set of records may be written again as part of restart processing whenever renodalization changes the number or position of plot data.

Restart records are written in sets beginning with a three word record containing 'restart' in Word 1. In most user processing of the RSTPLT file, these sets of restart records are skipped over and thus only a brief description of these records are given here. The records following the first three word records are written or read under control of information in the common block/comctl/. The first records are the individual common blocks needed for restart. SCDAP commons are written only if the problem specifies SCDAP components. Some common blocks do not vary with advancement time and are written only in the first restart block of a new problem or following a restart. Following the common blocks, dynamic blocks are written. A few dynamic blocks are written only in the first restart block or after a restart. The number and length of these records are problem dependent. A study of subroutine rstrec can show how these records are written and subroutine rrestf can show how they are read. Subroutine gninit initializes the portions of /comctl/ controlling writing/reading common blocks. Information on dynamic blocks is stored in /comctl/ as dynamic blocks are created, moved, or lengths changed.

After the first record is read, all records concerning plot data have one of the following four words. 'plotinf', 'plotalf', 'plotnum', or plotrec, as the first word. If the file is being processed only for plot information, any record not having one of these words can be ignored. Programming logic to process plot records for either RSTPLT or stripf files could be the following.

- Read the first record and perhaps check if it is a legitimate RSTPLT or stripf file from SCDAP.RELAP5.
- Then read the next record.
- If it is not a plotinf or plotrec file, read the next record.
- If it is a plotinf record, read following plotalf and plotnum records to obtain identification of plot variables.
- If it is a plotrec record, process the plot data.

Continue this process of reading records, testing the first word and proceeding according to its contects until the end of file is encountered. The program logic must provide for the possibility of more that one set of plotinf, plotalf, plotnum records. Study of the subroutine strip can show this process including how the plotalf and plotnum information is used to find desired information in the plot records.

The purpose of the strip option is to simplify obtaining plot information. Assuming an unformatted stripf file has been generated, a program can simply read and ignore the first four records, then just read the next records which will be only plot records and contain the plot information in the order requested in the input data to the strip run.

The sample output usually sent in a transmittal includes a strip run generating a formatted stripf file. Although this file is formatted and one record in the binary file might be written as several 80 column records, study of the sample output can help understand the binary files. The same information is written on either unformatted or formatted files. In formatted records, excluding the first record, each block of 80 column records (equivalent to a binary record) will have 'plotinf', 'plotalf', 'plotnum', or 'plotrec' as the first word.